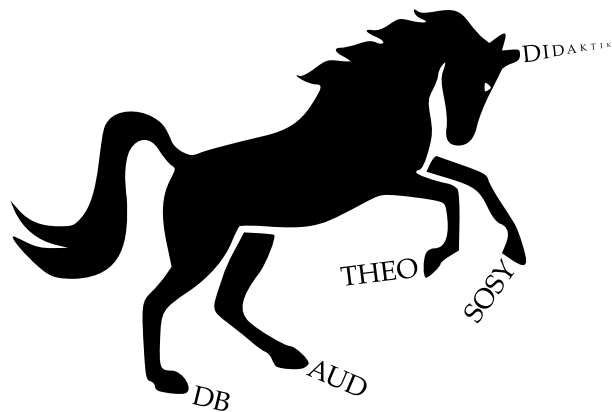


# Die komplette Sammlung

Alle Aufgaben



**Die Bschlangaul-Sammlung**

Hermine Bschlangaul and Friends

# Inhaltsverzeichnis

<b>I</b>	<b>Theoretische Informatik (THEO)</b>	<b>10</b>
<b>1</b>	<b>Reguläre Sprache</b>	<b>11</b>
	Grammatik aus Automat . . . . .	11
	Grammatik aus Automat . . . . .	11
	NEA-DEA-Äquivalenzklassen . . . . .	12
	NEA-DEA-Äquivalenzklassen . . . . .	12
	Noten . . . . .	14
	Noten . . . . .	14
	Reguläre Grammatik . . . . .	15
	Reguläre Grammatik . . . . .	15
	Übung zu regulären Grammatiken . . . . .	15
	Deterministischer endlicher Automat . . . . .	16
	Deterministischer endlicher Automat . . . . .	16
	Deterministisch endlicher Automat . . . . .	17
	Deterministisch endlicher Automat . . . . .	17
	Nichtdeterministischer endlicher Automat . . . . .	18
	Nichtdeterministischer endlicher Automat . . . . .	18
	Minimierungsalgorithmus . . . . .	18
	Minimierungsalgorithmus . . . . .	18
	Übung zur Minimalisierung . . . . .	20
	Übung zur Minimalisierung . . . . .	20
	Erweiterter Potenzmengenalgorithmus epsilon-NEA zum DEA . . . . .	21
	Erweiterter Potenzmengenalgorithmus $\epsilon$ -NEA zum DEA . . . . .	21
	Erweiterter Potenzmengenalgorithmus e-NEA zum DEA . . . . .	22
	Erweiterter Potenzmengenalgorithmus $\epsilon$ -NEA zum DEA . . . . .	22
	Potenzmengenalgorithmus . . . . .	23
	Potenzmengenalgorithmus . . . . .	23
	Potenzmengenalgorithmus . . . . .	24
	Potenzmengenalgorithmus . . . . .	24
	Pumping-Lemma . . . . .	25
	Pumping-Lemma . . . . .	25
	Pumping-Lemma . . . . .	26
	Pumping-Lemma . . . . .	26
	Pumping-Lemma . . . . .	27
	Pumping-Lemma . . . . .	27
	Übung zum Pumping-Lemma . . . . .	28
	Übung zum Pumping-Lemma . . . . .	28
	Aufgabe 4: Reguläre Ausdrücke in JAVA: . . . . .	29
	Aufgabe 4: Reguläre Ausdrücke in JAVA: Abitur 2018 Inf2. IV. 1. und Abitur 2015 Inf2. III. 2. . . . .	29

Reguläre Grammatik, reguläre Ausdrücke und DEA . . . . .	31
Reguläre Grammatik, reguläre Ausdrücke und DEA . . . . .	31
Übungen zu regulären Ausdrücken . . . . .	35
Übungen zu regulären Ausdrücken . . . . .	35
Reguläre Sprache in kontextfreier Sprache . . . . .	40
Reguläre Sprache in kontextfreier Sprache . . . . .	40
46115 / 2010 / Frühjahr / Thema 2 / Aufgabe 1 . . . . .	40
Aufgabe 1 . . . . .	40
46115 / 2015 / Frühjahr / Thema 1 / Aufgabe 1 . . . . .	42
Aufgabe 1 . . . . .	42
46115 / 2016 / Herbst / Thema 1 / Aufgabe 1 . . . . .	43
46115 / 2019 / Herbst / Thema 1 / Aufgabe 1 . . . . .	44
46115 / 2019 / Herbst / Thema 2 / Aufgabe 1 . . . . .	45
46115 / 2019 / Herbst / Thema 2 / Aufgabe 2 . . . . .	46
46115 / 2020 / Frühjahr / Thema 1 / Aufgabe 1 . . . . .	47
46115 / 2021 / Frühjahr / Thema 1 / Teilaufgabe 1 / Aufgabe 1 . . . . .	49
46115 / 2021 / Frühjahr / Thema 1 / Teilaufgabe 1 / Aufgabe 3 . . . . .	50
46115 / 2021 / Frühjahr / Thema 2 / Teilaufgabe 1 / Aufgabe 1 . . . . .	50
46115 / 2021 / Frühjahr / Thema 2 / Teilaufgabe 1 / Aufgabe 2 . . . . .	52
46115 / 2021 / Frühjahr / Thema 2 / Teilaufgabe 1 / Aufgabe 3 . . . . .	53
66115 / 2007 / Herbst / Thema 2 / Aufgabe 1 . . . . .	54
66115 / 2012 / Herbst / Thema 1 / Aufgabe 1 . . . . .	55
66115 / 2013 / Herbst / Thema 2 / Aufgabe 3 . . . . .	58
66115 / 2015 / Frühjahr / Thema 1 / Aufgabe 1 . . . . .	59
66115 / 2016 / Frühjahr / Thema 1 / Aufgabe 1 . . . . .	63
66115 / 2016 / Herbst / Thema 1 / Aufgabe 1 . . . . .	65
66115 / 2018 / Frühjahr / Thema 2 / Aufgabe 3 . . . . .	66
66115 / 2019 / Frühjahr / Thema 1 / Aufgabe 2 . . . . .	67
66115 / 2020 / Frühjahr / Thema 1 / Aufgabe 2 . . . . .	70
66115 / 2020 / Herbst / Thema 1 / Teilaufgabe 1 / Aufgabe 1 . . . . .	72
66115 / 2020 / Herbst / Thema 1 / Teilaufgabe 1 / Aufgabe 2 . . . . .	73
66115 / 2020 / Herbst / Thema 1 / Teilaufgabe 1 / Aufgabe 3 . . . . .	74
66115 / 2020 / Herbst / Thema 2 / Teilaufgabe 1 / Aufgabe 1 . . . . .	75
66115 / 2021 / Frühjahr / Thema 1 / Teilaufgabe 1 / Aufgabe 1 . . . . .	79
66115 / 2021 / Frühjahr / Thema 2 / Teilaufgabe 1 / Aufgabe 1 . . . . .	81

## 2 Kontextfreie Sprache 83

66115 / 2021 / Frühjahr / Thema 2 / Aufgabe 1 . . . . .	83
Ableitungen . . . . .	83
66115 / 2021 / Frühjahr / Thema 2 / Aufgabe 1 . . . . .	84
Ableitung und Parsebaum . . . . .	84
66115 / 2021 / Frühjahr / Thema 2 / Aufgabe 1 . . . . .	86
Klammerausdrücke . . . . .	86
66115 / 2021 / Frühjahr / Thema 2 / Aufgabe 1 . . . . .	88
Kontextfreie Grammatiken . . . . .	88
66115 / 2021 / Frühjahr / Thema 2 / Aufgabe 1 . . . . .	89

Kontextfreie Sprache . . . . .	89
66115 / 2021 / Frühjahr / Thema 2 / Aufgabe 1 . . . . .	92
Kontextfreie Sprache . . . . .	92
Übung . . . . .	92
Übung . . . . .	92
66115 / 2021 / Frühjahr / Thema 2 / Aufgabe 1 . . . . .	92
. . . . .	92
66115 / 2021 / Frühjahr / Thema 2 / Aufgabe 1 . . . . .	93
Wortproblem - CYK-Algorithmus . . . . .	93
66115 / 2021 / Frühjahr / Thema 2 / Aufgabe 1 . . . . .	95
CYK-Algorithmus . . . . .	95
66115 / 2021 / Frühjahr / Thema 2 / Aufgabe 1 . . . . .	95
CYK-Algorithmus <sup>1</sup> . . . . .	95
66115 / 2021 / Frühjahr / Thema 2 / Aufgabe 1 . . . . .	96
Chomsky-Normalform . . . . .	96
66115 / 2021 / Frühjahr / Thema 2 / Aufgabe 1 . . . . .	98
Kontextfreien Grammatiken in CNF . . . . .	98
66115 / 2021 / Frühjahr / Thema 2 / Aufgabe 1 . . . . .	103
Mindestens eine Eins . . . . .	103
66115 / 2021 / Frühjahr / Thema 2 / Aufgabe 1 . . . . .	103
Kontextfreie Grammatik . . . . .	103
66115 / 2021 / Frühjahr / Thema 2 / Aufgabe 1 . . . . .	105
Balancierte Klammern . . . . .	105
66115 / 2021 / Frühjahr / Thema 2 / Aufgabe 1 . . . . .	105
Kellerautomat . . . . .	105
66115 / 2021 / Frühjahr / Thema 2 / Aufgabe 1 . . . . .	106
Grammatik in Kellerautomat umwandeln . . . . .	106
66115 / 2021 / Frühjahr / Thema 2 / Aufgabe 1 . . . . .	108
66115 / 2021 / Frühjahr / Thema 2 / Aufgabe 1 . . . . .	110
PDA . . . . .	110
66115 / 2021 / Frühjahr / Thema 2 / Aufgabe 1 . . . . .	112
66115 / 2021 / Frühjahr / Thema 2 / Aufgabe 1 . . . . .	114
66115 / 2021 / Frühjahr / Thema 2 / Aufgabe 1 . . . . .	115
Pumping-Lemma . . . . .	115
66115 / 2021 / Frühjahr / Thema 2 / Aufgabe 1 . . . . .	116
Pumping-Lemma . . . . .	116
66115 / 2021 / Frühjahr / Thema 2 / Aufgabe 1 . . . . .	117
Pumping-Lemma . . . . .	117
46115 / 2019 / Herbst / Thema 1 / Aufgabe 2 . . . . .	119
46115 / 2021 / Frühjahr / Thema 1 / Teilaufgabe 1 / Aufgabe 2 . . . . .	121
66115 / 2012 / Frühjahr / Thema 1 / Aufgabe 3 . . . . .	121
66115 / 2012 / Frühjahr / Thema 1 / Aufgabe 4 . . . . .	123
66115 / 2013 / Frühjahr / Thema 1 / Aufgabe 2 . . . . .	124
66115 / 2016 / Frühjahr / Thema 1 / Aufgabe 2 . . . . .	126

---

<sup>1</sup><https://www.youtube.com/watch?v=Q5TvCyu4RUo>

66115 / 2017 / Frühjahr / Thema 2 / Aufgabe 2 . . . . .	127
66115 / 2017 / Herbst / Thema 1 / Aufgabe 2 . . . . .	130
66115 / 2017 / Herbst / Thema 2 / Aufgabe 5 . . . . .	131
66115 / 2018 / Herbst / Thema 1 / Aufgabe 3 . . . . .	132
66115 / 2019 / Frühjahr / Thema 1 / Aufgabe 3 . . . . .	133
66115 / 2020 / Frühjahr / Thema 1 / Aufgabe 3 . . . . .	135
66115 / 2020 / Frühjahr / Thema 2 / Aufgabe 3 . . . . .	137
66115 / 2020 / Herbst / Thema 2 / Teilaufgabe 1 / Aufgabe 2 . . . . .	138
66115 / 2021 / Frühjahr / Thema 1 / Teilaufgabe 1 / Aufgabe 2 . . . . .	138
66115 / 2021 / Frühjahr / Thema 2 / Teilaufgabe 1 / Aufgabe 2 . . . . .	139
<b>3 Kontextsensitive Sprache</b>	<b>142</b>
66115 / 2021 / Frühjahr / Thema 2 / Aufgabe 2 . . . . .	142
Kontextsensitive Grammatik . . . . .	142
66115 / 2021 / Frühjahr / Thema 2 / Aufgabe 2 . . . . .	142
Übung kontextsensitive Grammatiken [Seite 8]theo:fs:3 . . . . .	142
66115 / 2021 / Frühjahr / Thema 2 / Aufgabe 2 . . . . .	143
Komplement der Binärzahl . . . . .	143
Übung zu Turingmaschinen . . . . .	143
<b>4 Unbeschränkte Sprache</b>	<b>145</b>
66115 / 2021 / Frühjahr / Thema 2 / Aufgabe 2 . . . . .	145
Binärzahl dekrementieren . . . . .	145
66115 / 2021 / Frühjahr / Thema 2 / Aufgabe 2 . . . . .	146
Turing-Maschine zur Multiplikation mit 3 . . . . .	146
66115 / 2021 / Frühjahr / Thema 2 / Aufgabe 2 . . . . .	149
Turingmaschine mit folgender Übergangsfunktion . . . . .	149
66115 / 2021 / Frühjahr / Thema 2 / Aufgabe 2 . . . . .	153
Wörter umkehren auf der 2-Band-Turingmaschine . . . . .	153
66115 / 2021 / Frühjahr / Thema 2 / Aufgabe 2 . . . . .	153
Wörter umkehren auf dem Band der Turingmaschine . . . . .	153
66115 / 2021 / Frühjahr / Thema 2 / Aufgabe 2 . . . . .	155
Prädikat unäre Kodierung von $n$ und $m$ . . . . .	155
66115 / 2021 / Frühjahr / Thema 2 / Aufgabe 2 . . . . .	156
Ableitung einer Phrasenstrukturgrammatik . . . . .	156
46115 / 2013 / Frühjahr / Thema 1 / Aufgabe 4 . . . . .	157
Aufgabe 4 . . . . .	157
66115 / 2010 / Herbst / Thema 2 / Aufgabe 7 . . . . .	157
66115 / 2017 / Frühjahr / Thema 2 / Aufgabe 3 . . . . .	158
66115 / 2019 / Frühjahr / Thema 1 / Aufgabe 4 . . . . .	159
66115 / 2019 / Herbst / Thema 2 / Aufgabe 1 . . . . .	161
<b>5 Berechenbarkeit</b>	<b>163</b>
66115 / 2019 / Herbst / Thema 2 / Aufgabe 1 . . . . .	163
GOTO-Programme . . . . .	163
66115 / 2019 / Herbst / Thema 2 / Aufgabe 1 . . . . .	164

. . . . .	164
66115 / 2019 / Herbst / Thema 2 / Aufgabe 1 . . . . .	165
Vorlesungsaufgaben . . . . .	165
LOOP-Implementierung . . . . .	165
66115 / 2019 / Herbst / Thema 2 / Aufgabe 1 . . . . .	166
Primitiv-rekursiv . . . . .	166
66115 / 2019 / Herbst / Thema 2 / Aufgabe 1 . . . . .	167
Vorlesungsaufgaben . . . . .	167
Turing-berechenbar . . . . .	167
66115 / 2019 / Herbst / Thema 2 / Aufgabe 1 . . . . .	167
TURING-berechenbar . . . . .	167
66115 / 2019 / Herbst / Thema 2 / Aufgabe 1 . . . . .	168
Vorlesungsaufgaben WHILE-Programm . . . . .	168
66115 / 2019 / Herbst / Thema 2 / Aufgabe 1 . . . . .	168
66112 / 2002 / Frühjahr / Thema 2 / Aufgabe 5 . . . . .	170
66112 / 2004 / Herbst / Thema 2 / Aufgabe 3 . . . . .	170
66115 / 2015 / Frühjahr / Thema 2 / Aufgabe 4 . . . . .	170
66115 / 2016 / Frühjahr / Thema 2 / Aufgabe 2 . . . . .	170
66115 / 2016 / Herbst / Thema 2 / Aufgabe 3 . . . . .	172
66115 / 2020 / Frühjahr / Thema 2 / Aufgabe 4 . . . . .	173
<b>6 Entscheidbarkeit</b>	<b>175</b>
Post'sches Korrespondenzproblem (PCP) und Modifiziertes Post'sches Kor-	
respondenzproblem (MPCP) . . . . .	175
46115 / 2021 / Frühjahr / Thema 1 / Teilaufgabe 1 / Aufgabe 4 . . . . .	175
66115 / 2013 / Herbst / Thema 1 / Aufgabe 3 . . . . .	175
66115 / 2014 / Frühjahr / Thema 1 / Aufgabe 5 . . . . .	176
66115 / 2017 / Frühjahr / Thema 1 / Aufgabe 6 . . . . .	177
66115 / 2020 / Herbst / Thema 1 / Teilaufgabe 1 / Aufgabe 4 . . . . .	177
66115 / 2021 / Frühjahr / Thema 2 / Teilaufgabe 1 / Aufgabe 3 . . . . .	178
<b>7 Komplexitätstheorie</b>	<b>179</b>
66115 / 2021 / Frühjahr / Thema 2 / Aufgabe 3 . . . . .	179
Reduktion-Turingmaschine . . . . .	179
66115 / 2021 / Frühjahr / Thema 2 / Aufgabe 3 . . . . .	179
SAT-3SAT . . . . .	179
46115 / 2016 / Frühjahr / Thema 1 / Aufgabe 5 . . . . .	181
Aufgabe 5 . . . . .	181
46115 / 2016 / Frühjahr / Thema 2 / Aufgabe 2 . . . . .	181
Aufgabe 2 . . . . .	181
46115 / 2016 / Herbst / Thema 1 / Aufgabe 4 . . . . .	182
66115 / 2014 / Frühjahr / Thema 1 / Aufgabe 7 . . . . .	183
66115 / 2016 / Frühjahr / Thema 1 / Aufgabe 5 . . . . .	183
66115 / 2016 / Frühjahr / Thema 2 / Aufgabe 3 . . . . .	184
66115 / 2017 / Herbst / Thema 1 / Aufgabe 3 . . . . .	185
66115 / 2018 / Herbst / Thema 2 / Aufgabe 4 . . . . .	186

66115 / 2019 / Frühjahr / Thema 1 / Aufgabe 5 . . . . .	187
66115 / 2019 / Herbst / Thema 1 / Aufgabe 4 . . . . .	188
66115 / 2020 / Herbst / Thema 1 / Teilaufgabe 1 / Aufgabe 5 . . . . .	189
66115 / 2020 / Herbst / Thema 2 / Teilaufgabe 2 / Aufgabe 2 . . . . .	189
66115 / 2021 / Frühjahr / Thema 2 / Teilaufgabe 1 / Aufgabe 4 . . . . .	190

## 8 Sonstige 192

## 9 SOSY 193

66115 / 2021 / Frühjahr / Thema 2 / Aufgabe 4 . . . . .	193
66115 / 2021 / Frühjahr / Thema 2 / Aufgabe 4 . . . . .	193
66115 / 2021 / Frühjahr / Thema 2 / Aufgabe 4 . . . . .	194
66115 / 2021 / Frühjahr / Thema 2 / Aufgabe 4 . . . . .	198
66115 / 2021 / Frühjahr / Thema 2 / Aufgabe 4 . . . . .	203
66115 / 2021 / Frühjahr / Thema 2 / Aufgabe 4 . . . . .	204
66115 / 2021 / Frühjahr / Thema 2 / Aufgabe 4 . . . . .	204
66115 / 2021 / Frühjahr / Thema 2 / Aufgabe 4 . . . . .	205
66115 / 2021 / Frühjahr / Thema 2 / Aufgabe 4 . . . . .	210
66115 / 2021 / Frühjahr / Thema 2 / Aufgabe 4 . . . . .	213
66115 / 2021 / Frühjahr / Thema 2 / Aufgabe 4 . . . . .	216
66115 / 2021 / Frühjahr / Thema 2 / Aufgabe 4 . . . . .	217
66115 / 2021 / Frühjahr / Thema 2 / Aufgabe 4 . . . . .	219
66115 / 2021 / Frühjahr / Thema 2 / Aufgabe 4 . . . . .	219
66115 / 2021 / Frühjahr / Thema 2 / Aufgabe 4 . . . . .	222
66115 / 2021 / Frühjahr / Thema 2 / Aufgabe 4 . . . . .	229
66115 / 2021 / Frühjahr / Thema 2 / Aufgabe 4 . . . . .	231
66115 / 2021 / Frühjahr / Thema 2 / Aufgabe 4 . . . . .	239
66115 / 2021 / Frühjahr / Thema 2 / Aufgabe 4 . . . . .	240
66115 / 2021 / Frühjahr / Thema 2 / Aufgabe 4 . . . . .	242
66115 / 2021 / Frühjahr / Thema 2 / Aufgabe 4 . . . . .	243
66115 / 2021 / Frühjahr / Thema 2 / Aufgabe 4 . . . . .	244
66115 / 2021 / Frühjahr / Thema 2 / Aufgabe 4 . . . . .	245
66115 / 2021 / Frühjahr / Thema 2 / Aufgabe 4 . . . . .	248
66115 / 2021 / Frühjahr / Thema 2 / Aufgabe 4 . . . . .	250
66115 / 2021 / Frühjahr / Thema 2 / Aufgabe 4 . . . . .	252
66115 / 2021 / Frühjahr / Thema 2 / Aufgabe 4 . . . . .	253
66115 / 2021 / Frühjahr / Thema 2 / Aufgabe 4 . . . . .	253
66115 / 2021 / Frühjahr / Thema 2 / Aufgabe 4 . . . . .	254
66115 / 2021 / Frühjahr / Thema 2 / Aufgabe 4 . . . . .	256
66115 / 2021 / Frühjahr / Thema 2 / Aufgabe 4 . . . . .	258
66115 / 2021 / Frühjahr / Thema 2 / Aufgabe 4 . . . . .	259
66115 / 2021 / Frühjahr / Thema 2 / Aufgabe 4 . . . . .	262
66115 / 2021 / Frühjahr / Thema 2 / Aufgabe 4 . . . . .	262
46115 / 2015 / Herbst / Thema 2 / Aufgabe 4 . . . . .	264
46116 / 2010 / Frühjahr / Thema 1 / Aufgabe 1 . . . . .	265
46116 / 2011 / Frühjahr / Thema 1 / Teilaufgabe 2 / Aufgabe 1 . . . . .	268

Firmenstruktur . . . . .	268
46116 / 2012 / Frühjahr / Thema 2 / Teilaufgabe 2 / Aufgabe 1 . . . . .	271
Hotel-Verwaltung . . . . .	271
46116 / 2013 / Frühjahr / Thema 1 / Teilaufgabe 1 / Aufgabe 2 . . . . .	272
46116 / 2013 / Frühjahr / Thema 2 / Teilaufgabe 1 / Aufgabe 1 . . . . .	274
Aufgabe 1 . . . . .	274
46116 / 2014 / Frühjahr / Thema 2 / Teilaufgabe 1 / Aufgabe 1 . . . . .	276
46116 / 2014 / Herbst / Thema 2 / Teilaufgabe 1 / Aufgabe 1 . . . . .	277
Aufgabe 1: Allgemeine SWT, Vorgehensmodelle und Requirements Enginee- ring . . . . .	277
46116 / 2014 / Herbst / Thema 2 / Teilaufgabe 1 / Aufgabe 2 . . . . .	279
46116 / 2014 / Herbst / Thema 2 / Teilaufgabe 1 / Aufgabe 3 . . . . .	282
46116 / 2015 / Frühjahr / Thema 1 / Teilaufgabe 1 / Aufgabe 3 . . . . .	284
46116 / 2015 / Herbst / Thema 1 / Teilaufgabe 1 / Aufgabe 3 . . . . .	285
46116 / 2015 / Herbst / Thema 2 / Teilaufgabe 1 / Aufgabe 2 . . . . .	287
46116 / 2015 / Herbst / Thema 2 / Teilaufgabe 1 / Aufgabe 3 . . . . .	291
46116 / 2016 / Frühjahr / Thema 2 / Teilaufgabe 1 / Aufgabe 2 . . . . .	297
46116 / 2016 / Herbst / Thema 2 / Teilaufgabe 1 / Aufgabe 4 . . . . .	299
46116 / 2017 / Frühjahr / Thema 1 / Teilaufgabe 1 / Aufgabe 2 . . . . .	302
46116 / 2017 / Frühjahr / Thema 1 / Teilaufgabe 1 / Aufgabe 3 . . . . .	303
46116 / 2017 / Frühjahr / Thema 1 / Teilaufgabe 1 / Aufgabe 5 . . . . .	304
46116 / 2017 / Herbst / Thema 1 / Teilaufgabe 1 / Aufgabe 4 . . . . .	305
46116 / 2017 / Herbst / Thema 2 / Teilaufgabe 1 / Aufgabe 3 . . . . .	307
46116 / 2018 / Frühjahr / Thema 1 / Teilaufgabe 1 / Aufgabe 3 . . . . .	307
46116 / 2018 / Herbst / Thema 1 / Teilaufgabe 1 / Aufgabe 3 . . . . .	312
66112 / 2002 / Herbst / Thema 1 / Aufgabe 4 . . . . .	314
66112 / 2003 / Herbst / Thema 2 / Aufgabe 5 . . . . .	318
66112 / 2005 / Frühjahr / Thema 1 / Aufgabe 1 . . . . .	319
66115 / 2014 / Frühjahr / Thema 1 / Aufgabe 1 . . . . .	322
66115 / 2017 / Frühjahr / Thema 1 / Aufgabe 4 . . . . .	325
66116 / 2012 / Herbst / Thema 2 / Teilaufgabe 2 / Aufgabe 2 . . . . .	327
66116 / 2013 / Herbst / Thema 1 / Teilaufgabe 2 / Aufgabe 3 . . . . .	330
66116 / 2014 / Herbst / Thema 2 / Teilaufgabe 2 / Aufgabe 2 . . . . .	333
66116 / 2014 / Herbst / Thema 2 / Teilaufgabe 2 / Aufgabe 3 . . . . .	333
66116 / 2015 / Frühjahr / Thema 1 / Teilaufgabe 2 / Aufgabe 2 . . . . .	335
66116 / 2015 / Frühjahr / Thema 1 / Teilaufgabe 2 / Aufgabe 3 . . . . .	336
66116 / 2015 / Frühjahr / Thema 2 / Teilaufgabe 2 / Aufgabe 2 . . . . .	340
66116 / 2015 / Frühjahr / Thema 2 / Teilaufgabe 2 / Aufgabe 3 . . . . .	343
66116 / 2015 / Herbst / Thema 2 / Teilaufgabe 2 / Aufgabe 3 . . . . .	345
66116 / 2016 / Frühjahr / Thema 1 / Teilaufgabe 2 / Aufgabe 2 . . . . .	349
66116 / 2016 / Frühjahr / Thema 2 / Teilaufgabe 2 / Aufgabe 4 . . . . .	350
66116 / 2016 / Herbst / Thema 1 / Teilaufgabe 2 / Aufgabe 1 . . . . .	351
66116 / 2016 / Herbst / Thema 1 / Teilaufgabe 2 / Aufgabe 2 . . . . .	354
66116 / 2016 / Herbst / Thema 1 / Teilaufgabe 2 / Aufgabe 3 . . . . .	357
66116 / 2016 / Herbst / Thema 2 / Teilaufgabe 2 / Aufgabe 1 . . . . .	360
66116 / 2017 / Frühjahr / Thema 2 / Teilaufgabe 2 / Aufgabe 1 . . . . .	362



66116 / 2017 / Frühjahr / Thema 2 / Teilaufgabe 2 / Aufgabe 4 . . . . .	366
66116 / 2017 / Herbst / Thema 1 / Teilaufgabe 2 / Aufgabe 4 . . . . .	369
66116 / 2017 / Herbst / Thema 2 / Teilaufgabe 2 / Aufgabe 1 . . . . .	373
66116 / 2017 / Herbst / Thema 2 / Teilaufgabe 2 / Aufgabe 2 . . . . .	374
66116 / 2018 / Frühjahr / Thema 2 / Teilaufgabe 2 / Aufgabe 1 . . . . .	375
66116 / 2018 / Frühjahr / Thema 2 / Teilaufgabe 2 / Aufgabe 2 . . . . .	377
66116 / 2018 / Herbst / Thema 1 / Teilaufgabe 1 / Aufgabe 2 . . . . .	380
66116 / 2018 / Herbst / Thema 2 / Teilaufgabe 1 / Aufgabe 1 . . . . .	384
66116 / 2019 / Frühjahr / Thema 1 / Teilaufgabe 2 / Aufgabe 1 . . . . .	385
66116 / 2019 / Frühjahr / Thema 1 / Teilaufgabe 2 / Aufgabe 2 . . . . .	386
66116 / 2019 / Frühjahr / Thema 1 / Teilaufgabe 2 / Aufgabe 3 . . . . .	387
66116 / 2019 / Frühjahr / Thema 1 / Teilaufgabe 2 / Aufgabe 4 . . . . .	388
66116 / 2019 / Frühjahr / Thema 2 / Teilaufgabe 2 / Aufgabe 1 . . . . .	388
66116 / 2019 / Herbst / Thema 1 / Teilaufgabe 1 / Aufgabe 1 . . . . .	390
66116 / 2019 / Herbst / Thema 1 / Teilaufgabe 1 / Aufgabe 2 . . . . .	391
66116 / 2019 / Herbst / Thema 1 / Teilaufgabe 1 / Aufgabe 3 . . . . .	391
66116 / 2019 / Herbst / Thema 1 / Teilaufgabe 1 / Aufgabe 4 . . . . .	393
66116 / 2019 / Herbst / Thema 2 / Teilaufgabe 1 / Aufgabe 1 . . . . .	399
66116 / 2019 / Herbst / Thema 2 / Teilaufgabe 1 / Aufgabe 3 . . . . .	404
66116 / 2020 / Frühjahr / Thema 1 / Teilaufgabe 1 / Aufgabe 1 . . . . .	405
66116 / 2020 / Herbst / Thema 1 / Teilaufgabe 1 / Aufgabe 1 . . . . .	406
66116 / 2020 / Herbst / Thema 1 / Teilaufgabe 1 / Aufgabe 2 . . . . .	409
66116 / 2020 / Herbst / Thema 1 / Teilaufgabe 1 / Aufgabe 3 . . . . .	413
66116 / 2020 / Herbst / Thema 1 / Teilaufgabe 1 / Aufgabe 4 . . . . .	416
66116 / 2020 / Herbst / Thema 1 / Teilaufgabe 1 / Aufgabe 5 . . . . .	421
66116 / 2020 / Herbst / Thema 2 / Teilaufgabe 1 / Aufgabe 1 . . . . .	422
66116 / 2020 / Herbst / Thema 2 / Teilaufgabe 1 / Aufgabe 2 . . . . .	424
66116 / 2020 / Herbst / Thema 2 / Teilaufgabe 1 / Aufgabe 4 . . . . .	425
66116 / 2020 / Herbst / Thema 2 / Teilaufgabe 1 / Aufgabe 5 . . . . .	427
66116 / 2020 / Herbst / Thema 2 / Teilaufgabe 1 / Aufgabe 6 . . . . .	430
66116 / 2021 / Frühjahr / Thema 1 / Teilaufgabe 1 / Aufgabe 10 . . . . .	431
66116 / 2021 / Frühjahr / Thema 1 / Teilaufgabe 1 / Aufgabe 11 . . . . .	431
66116 / 2021 / Frühjahr / Thema 1 / Teilaufgabe 1 / Aufgabe 12 . . . . .	432
66116 / 2021 / Frühjahr / Thema 1 / Teilaufgabe 1 / Aufgabe 3 . . . . .	433
66116 / 2021 / Frühjahr / Thema 1 / Teilaufgabe 1 / Aufgabe 6 . . . . .	434
66116 / 2021 / Frühjahr / Thema 1 / Teilaufgabe 1 / Aufgabe 7 . . . . .	435
66116 / 2021 / Frühjahr / Thema 1 / Teilaufgabe 1 / Aufgabe 8 . . . . .	435
66116 / 2021 / Frühjahr / Thema 1 / Teilaufgabe 1 / Aufgabe 9 . . . . .	436
66116 / 2021 / Frühjahr / Thema 2 / Teilaufgabe 1 / Aufgabe 1 . . . . .	436
66116 / 2021 / Frühjahr / Thema 2 / Teilaufgabe 1 / Aufgabe 2 . . . . .	441
66116 / 2021 / Frühjahr / Thema 2 / Teilaufgabe 1 / Aufgabe 4 . . . . .	444

## **Teil I**

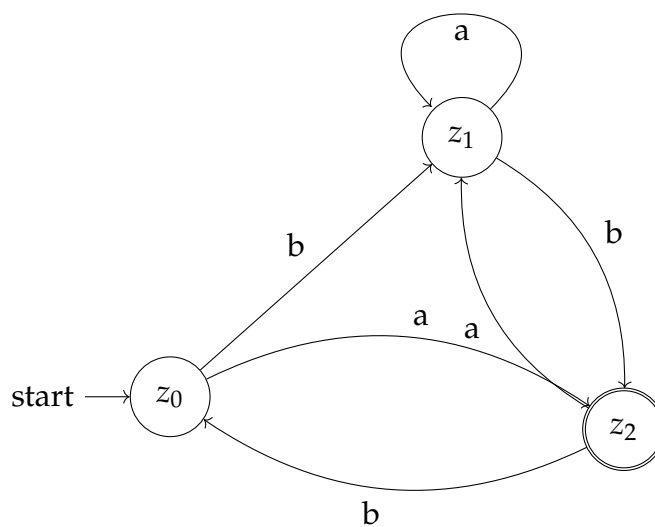
# **Theoretische Informatik (THEO)**

# Reguläre Sprache

## Grammatik aus Automat

## Grammatik aus Automat

Sei  $A = (\{z_0, z_1, z_2\}, \{a, b\}, \delta, \{z_2\}, z_0)$  ein endlicher Automat. Die Übergangsfunktion sei wie in dem unten abgebildeten Diagramm definiert.



Der Automat auf [flaci.com](http://flaci.com) (FLACI: Formale Sprachen, abstrakte Automaten, Compiler und Interpreter) Ein Projekt der Hochschule Zittau/Görlitz und der Pädagogischen Hochschule Schwyz: [flaci.com/Apk0iyqyg](http://flaci.com/Apk0iyqyg)

(a) Gebe eine reguläre Grammatik  $G$  an, sodass  $L(G) = L(M)$  gilt.

Lösungsvorschlag

$G = (\{Z_0, Z_1, Z_2\}, \{a, b\}, P, Z_0)$  mit folgender Produktionsmenge

$P = \{$

$Z_0 \rightarrow bZ_1 \mid aZ_2$

$Z_1 \rightarrow aZ_1 \mid bZ_2$

$Z_2 \rightarrow bZ_0 \mid aZ_1 \mid \varepsilon$

$\}$

(b) Überlegen Sie sich ein systematisches Verfahren, um einen deterministischen endlichen Automaten in eine reguläre Grammatik umzuwandeln.

Analog zu obigem Beispiel folgender Algorithmus benutzt werden:

- Setze  $V = \{Z_0, Z_1, \dots, Z_n\}$  und  $S$  auf den Startzustand des Automaten.
- Für jeden Übergang  $\delta(Z_i, a) = Z_j$  füge die Produktion  $\{Z_i \rightarrow aZ_j\}$  zu  $P$  hinzu.
- Für jeden Zustand  $Z_i \in Z$  füge die Produktion  $\{Z_i \rightarrow \varepsilon\}$  zu  $P$  dazu.

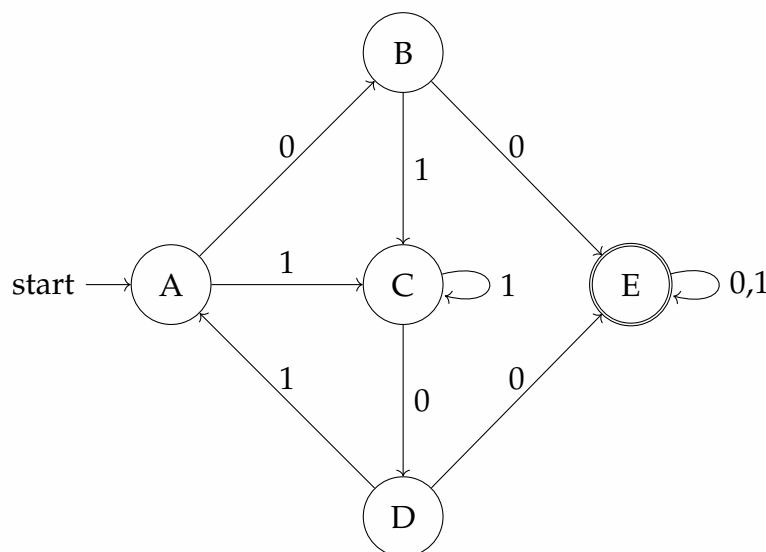
## NEA-DEA-Äquivalenzklassen

## NEA-DEA-Äquivalenzklassen

Gegeben ist der deterministische endliche Automat  $A = (\{A, B, C, D, E\}, \{0, 1\}, \delta, \{E\}, A)$ .

$\delta$	0	1
A	B	C
B	E	C
C	D	C
D	E	A
E	E	E

- (a) Minimieren Sie den Automaten mit dem bekannten Minimierungsalgorithmus. Dokumentieren Sie die Schritte geeignet.



Der Automat auf [flaci.com](http://flaci.com) (FLACI: Formale Sprachen, abstrakte Automaten, Compiler und Interpreter) Ein Projekt der Hochschule Zittau/Görlitz und der Pädagogischen Hochschule Schwyz: [flaci.com/A5amu40wc](http://flaci.com/A5amu40wc)

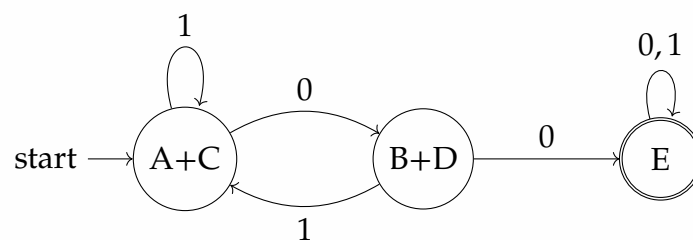
A	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$
B	$x_2$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$
C		$x_2$	$\emptyset$	$\emptyset$	$\emptyset$
D	$x_2$		$x_2$	$\emptyset$	$\emptyset$
E	$x_1$	$x_1$	$x_1$	$x_1$	$\emptyset$
	A	B	C	D	E

- $x_1$  Paar aus End-/ Nicht-Endzustand kann nicht äquivalent sein.  
 $x_2$  Test, ob man mit der Eingabe zu einem bereits markiertem Paar kommt.  
 $x_3$  In weiteren Iterationen markierte Zustände.  
 $x_4$  ...

### Übergangstabelle

Zustandspaar	0	1
(A, B)	(B, E) $x_2$	(C, C)
(A, C)	(B, D)	(C, C)
(A, D)	(B, E) $x_2$	(C, A)
(B, C)	(E, D) $x_2$	(C, C)
(B, D)	(E, E)	(C, A)
(C, D)	(D, E) $x_2$	(C, A)

### Minimiert



Der Automat auf [flaci.com](http://flaci.com) (FLACI: Formale Sprachen, abstrakte Automaten, Compiler und Interpreter) Ein Projekt der Hochschule Zittau/Görlitz und der Pädagogischen Hochschule Schwyz: [flaci.com/Ara57j4oa](http://flaci.com/Ara57j4oa)

(b) Geben Sie einen regulären Ausdruck für die erkannte Sprache an.

$$r = (0|1)^*00(0|1)^*$$

- (c) Geben Sie die Äquivalenzklassen der Myhill-Nerode-Äquivalenz der Sprache durch reguläre Ausdrücke an.

Die Äquivalenzklassen lauten:  $[A, C], [B, D], [E]$

$$r_A = (1^*(01)^*)^*$$

$$r_B = (1^*(01)^*)^*0$$

$$r_C = r$$

## Noten

## Noten

Der Viervierteltakt ist vor allem in der Unterhaltungsmusik die weitaus häufigste Taktart. Viervierteltakt bedeutet, dass ein Takt eine Länge von vier Vierteln hat. Das Symbol ♩ bedeutet eine Viertelnote. Eine halbe Note ♪ hat den Wert von zwei Vierteln, eine punktierte halbe Note ♪. den Wert von drei Vierteln und eine ganze Note ♩ den Wert von vier Vierteln. Andere Notenwerte sollen hier nicht vorkommen.

Die Prüfsoftware eines Notenverlages stellt eine Methode bereit, die testet, ob die Notenwerte eines Taktes tatsächlich vier Viertel ergeben.

Durch das Alphabet

$$\Sigma = \{ \text{♩}, \text{♪}, \text{♪.}, \text{♩} \},$$

die Menge der Nichtterminale

$$V = \{ \langle \text{Takt} \rangle; \langle 3/4 \rangle; \langle 1/2 \rangle \},$$

das Startsymbol  $\langle \text{Takt} \rangle$  sowie die Produktionsregeln

$$R_1: \langle \text{Takt} \rangle \rightarrow \text{♩} \mid \langle 3/4 \rangle \text{ ♩}$$

$$R_2: \langle 3/4 \rangle \rightarrow \text{♪.} \mid \langle 1/2 \rangle \text{ ♩}$$

$$R_3: \langle 1/2 \rangle \rightarrow \text{♪} \mid \text{♪ ♩}$$

ist eine Grammatik für eine formale Sprache  $S$  gegeben.

- (a) Entscheide begründet, ob die Zeichenketten ♩♩♩♩ und ♩♪♪ zu  $S$  gehören. Gebe eine weitere Zeichenkette an, die zwar einen Viervierteltakt darstellt, aber nicht zu  $S$  gehört.

- (b) Ergänze die oben angegebenen Produktionsregeln so, dass jeder mit dem Alphabet  $\Sigma$  mögliche Viervierteltakt dargestellt werden kann.
- (c) Mit  $T_4$  wird die formale Sprache bezeichnet, die genau alle Viervierteltakte enthält, die mit den Zeichen aus  $\Sigma$  gebildet werden können. Verwende für die weiteren Teilaufgaben anstelle der Notensymbole Buchstaben gemäß folgender Tabelle:  
Zeichne das Zustandsübergangsdiagramm eines erkennenden endlichen Automaten, der genau  $T_4$  akzeptiert.
- (d) Entwerfe eine Implementierung des Automaten aus Teilaufgabe c in Java. Dabei soll es u. a. eine Methode `istViervierteltakt(eingabe)` geben, die überprüft, ob die übergebene Zeichenkette `eingabe` den Vorgaben für einen Viervierteltakt entspricht, und einen entsprechenden Wahrheitswert zurückgibt. Dazu ruft sie für jedes Zeichen der Eingabe jeweils die Methode `zustandWechseln()` auf.

## Reguläre Grammatik

## Reguläre Grammatik

## Übung zu regulären Grammatiken

Gegeben ist eine Sprache  $L \subset \Sigma^*$  mit  $\Sigma = \{a, b\}$ . Zu der Sprache  $L$  gehören alle Wörter, die die Zeichenfolge *abba* beinhalten.

- (a) Gib eine Grammatik an, die diese Sprache erzeugt.

Lösungsvorschlag

$G = (V, \Sigma, P, S)$  mit  $\Sigma = \{a, b\}$ ,  $S = S$ ,  $V = \{S, A, B, C, D\}$

Tipp: Die Produktionsregeln so entwerfen, dass zuerst das Wort „abba“ erkannt wird, dann die Regeln entwerfen.

$P = \{$

$S \rightarrow aA \mid aS \mid bS$

$A \rightarrow bB$

$B \rightarrow bC$

$C \rightarrow aD$

$D \rightarrow aD \mid bD \mid \varepsilon$

$\}$

Andere Möglichkeit:

$P = \{$

$$\begin{aligned}
 S &\rightarrow aA \mid aS \mid bS \\
 A &\rightarrow bB \\
 B &\rightarrow bC \\
 C &\rightarrow aD \mid a \\
 D &\rightarrow aD \mid bD \mid a \mid b
 \end{aligned}$$

}

Nicht erlaubt in regulärer Grammtik:

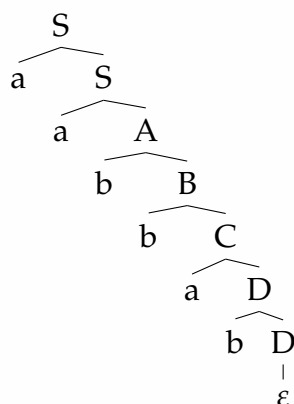
$$P = \{$$

$$S \rightarrow abbaA$$

}

(b) Gib eine Ableitung/Syntaxbaum zu deiner Grammatik für das Wort aabbab an.

Lösungsvorschlag

$$S \rightarrow aS \rightarrow aaA \rightarrow aabB \rightarrow aabbC \rightarrow aabbaD \rightarrow aabbabD \rightarrow aabbab$$


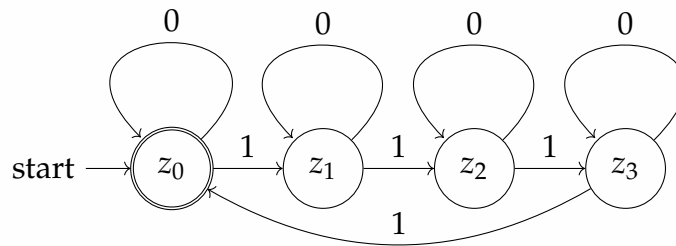
## Deterministischer endlicher Automat

## Deterministischer endlicher Automat

Geben Sie einen DFA über dem Alphabet  $\Sigma = \{0, 1\}$  an, der die folgende Sprache erkennt: Die Menge aller Zeichenketten, die eine durch 4 teilbare Anzahl von Einsen besitzt.

Lösungsvorschlag





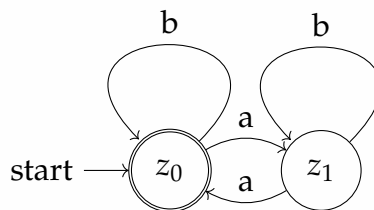
## Deterministisch endlicher Automat

## Deterministisch endlicher Automat

Stellen Sie einen Automaten zu den folgenden Sprachen ( $\Sigma = \{a, b\}$ ) auf:

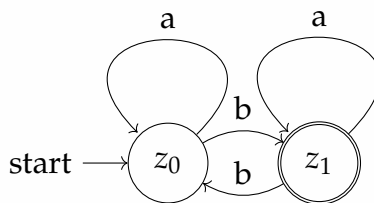
- (a)  $L_1 = \{x \mid x \text{ beinhaltet eine gerade Anzahl von } a\}$

Lösungsvorschlag



- (b)  $L_2 = \{x \mid x \text{ beinhaltet eine ungerade Anzahl von } b\}$

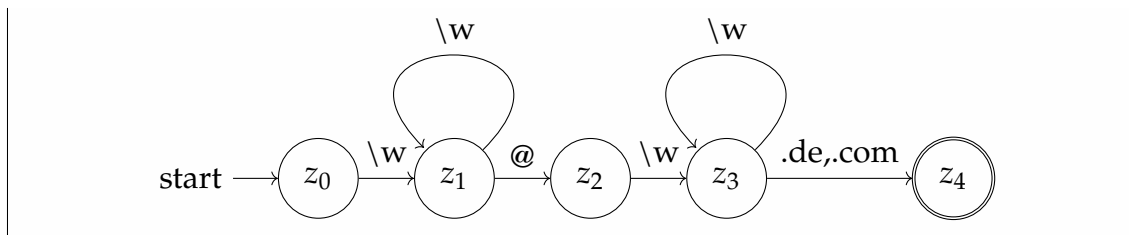
Lösungsvorschlag



- (c) Geben Sie einen DEA an, der eine syntaktisch gültige E-Mail-Adresse erkennt. (mindestens 1 Zeichen (Groß-/Kleinbuchstabe oder Zahl) vor dem @; mindestens 1 Zeichen (Groß-/Kleinbuchstabe oder Zahl) nach dem @; alle E-Mail-Adressen sollen auf .de oder .com enden.

Lösungsvorschlag

Es müsste noch der Trap-Zustand hinzugefügt werden.



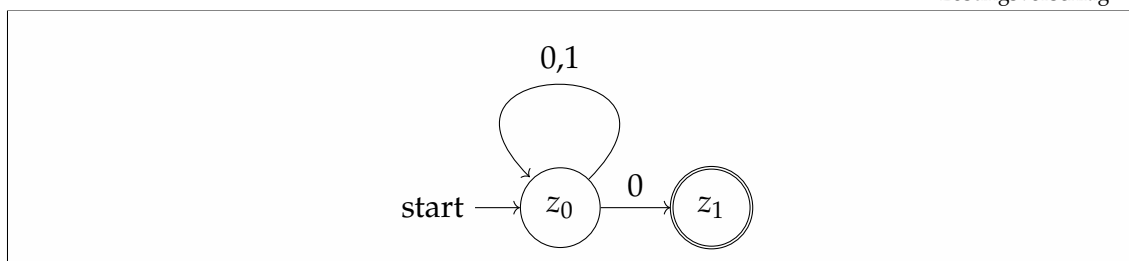
Nichtdeterministisch  
endlicher Automat (NEA)  
Minimierungsalgorithmus

## Nichtdeterministischer endlicher Automat

## Nichtdeterministischer endlicher Automat

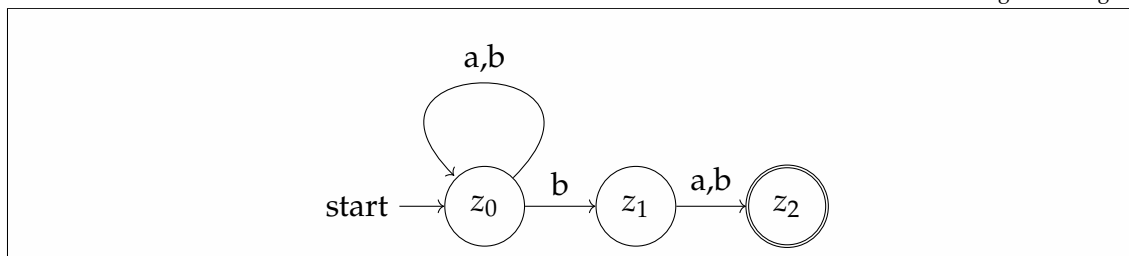
- (a) Stellen Sie einen nichtdeterministischen endlichen Automaten auf, der alle durch 2 teilbaren Binärzahlen (letztes Wort ist 0) akzeptiert.

Lösungsvorschlag



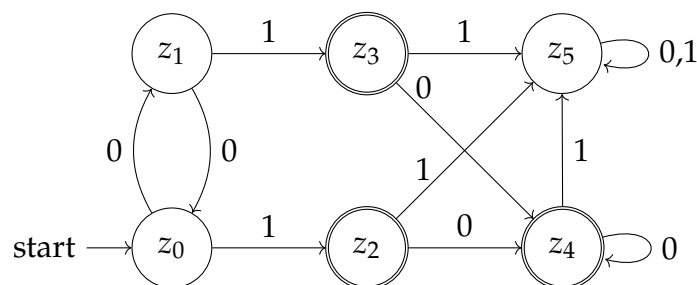
- (b) Stellen Sie einen NEA auf, der alle Wörter über einem Alphabet  $\Sigma = \{a, b\}$  akzeptiert, die als vorletztes Zeichen ein  $b$  besitzen.

Lösungsvorschlag



## Minimierungsalgorithmus

## Minimierungsalgorithmus



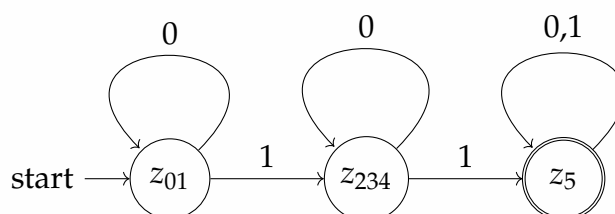
$z_0$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$
$z_1$		$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$
$z_2$	$x_1$	$x_1$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$
$z_3$	$x_1$	$x_1$		$\emptyset$	$\emptyset$	$\emptyset$
$z_4$	$x_1$	$x_1$			$\emptyset$	$\emptyset$
$z_5$	$x_2$	$x_2$	$x_1$	$x_1$	$x_1$	$\emptyset$
	$z_0$	$z_1$	$z_2$	$z_3$	$z_4$	$z_5$

- $x_1$  Paar aus End-/ Nicht-Endzustand kann nicht äquivalent sein.  
 $x_2$  Test, ob man mit der Eingabe zu einem bereits markiertem Paar kommt.  
 $x_3$  In weiteren Iterationen markierte Zustände.  
 $x_4$  ...

### Übergangstabelle

Zustandspaar	0	1
$(z_0, z_1)$	$(z_1, z_0)$	$(z_2, z_3)$
$(z_0, z_5)$	$(z_1, z_5)$	$(z_2, z_5) \ x_2$
$(z_1, z_5)$	$(z_0, z_5)$	$(z_3, z_5) \ x_2$
$(z_2, z_3)$	$(z_4, z_4)$	$(z_5, z_5)$
$(z_2, z_4)$	$(z_4, z_4)$	$(z_5, z_5)$
$(z_3, z_4)$	$(z_4, z_4)$	$(z_5, z_5)$

$(z_2, z_3)$ ,  $(z_2, z_4)$  und  $(z_3, z_4)$  können zu einem Zustand verschmolzen werden, weil sie alle drei bei der Eingabe von 0 zu  $(z_4, z_4)$  und bei 1 zu  $(z_5, z_5)$  werden.  $z_5$  kann nicht verschmolzen werden, weil er in der Tabelle markiert ist.

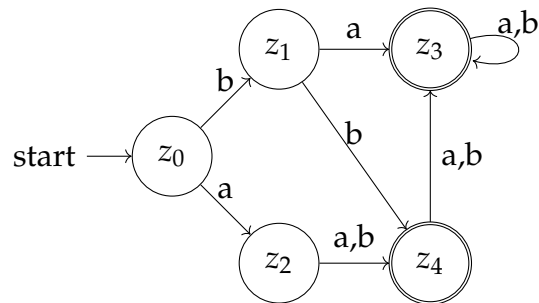


<sup>1</sup><https://studyflix.de/informatik/dea-minimieren-1212>

## Übung zur Minimalisierung

## Übung zur Minimalisierung

Minimalisiere den gegebenen DEA:



Der Automat auf flaci.com (FLACI: Formale Sprachen, abstrakte Automaten, Compiler und Interpreter) Ein Projekt der Hochschule Zittau/Görlitz und der Pädagogischen Hochschule Schwyz: [flaci.com/Apm4e9nk7](http://flaci.com/Apm4e9nk7)

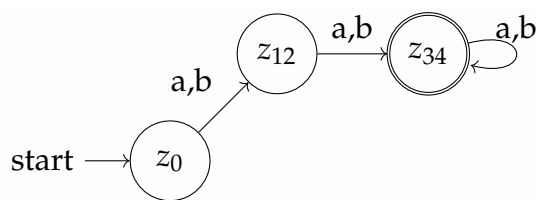
Lösungsvorschlag

z <sub>0</sub>	∅	∅	∅	∅	∅
z <sub>1</sub>	x <sub>2</sub>	∅	∅	∅	∅
z <sub>2</sub>	x <sub>2</sub>		∅	∅	∅
z <sub>3</sub>	x <sub>1</sub>	x <sub>1</sub>	x <sub>1</sub>	∅	∅
z <sub>4</sub>	x <sub>1</sub>	x <sub>1</sub>	x <sub>1</sub>		∅
	z <sub>0</sub>	z <sub>1</sub>	z <sub>2</sub>	z <sub>3</sub>	z <sub>4</sub>

- x<sub>1</sub> Paar aus End-/ Nicht-Endzustand kann nicht äquivalent sein.
- x<sub>2</sub> Test, ob man mit der Eingabe zu einem bereits markiertem Paar kommt.
- x<sub>3</sub> In weiteren Iterationen markierte Zustände.
- x<sub>4</sub> ...

### Übergangstabelle

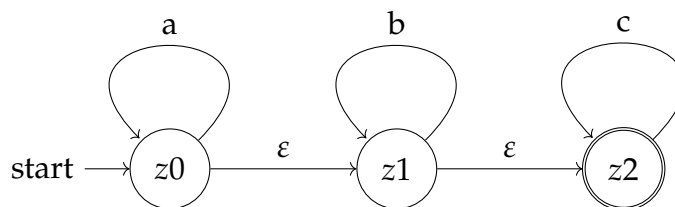
Zustandspaar	a	b
(z <sub>0</sub> , z <sub>1</sub> )	(z <sub>2</sub> , z <sub>3</sub> ) x <sub>2</sub>	(z <sub>1</sub> , z <sub>4</sub> )
(z <sub>0</sub> , z <sub>2</sub> )	(z <sub>2</sub> , z <sub>4</sub> ) x <sub>2</sub>	(z <sub>1</sub> , z <sub>4</sub> )
(z <sub>1</sub> , z <sub>2</sub> )	(z <sub>3</sub> , z <sub>4</sub> )	(z <sub>4</sub> , z <sub>4</sub> )
(z <sub>3</sub> , z <sub>4</sub> )	(z <sub>3</sub> , z <sub>3</sub> )	(z <sub>3</sub> , z <sub>3</sub> )



Der Automat auf flaci.com (FLACI: Formale Sprachen, abstrakte Automaten, Compiler und Interpreter) Ein Projekt der Hochschule Zittau/Görlitz und der Pädagogischen Hochschule Schwyz: flaci.com/Aib87m3wc

## Erweiterter Potenzmengenalgorithmus epsilon-NEA zum DEA

## Erweiterter Potenzmengenalgorithmus $\varepsilon$ -NEA zum DEA



- (a) Welche Sprache akzeptiert dieser Automat? Beschreiben Sie in Worten und stellen Sie einen regulären Ausdruck sowie eine Grammatik hierfür auf.

Lösungsvorschlag

**in Worten** Das Alphabet besteht aus  $a, b, c$ . Am Anfang stehen 0 oder beliebig viele  $a$ 's, dann kommen 0 oder beliebig viele  $b$ 's und dann 0 oder beliebig viele  $c$ 's.

**Regulärer Ausdruck**  $a^*b^*c^*$

**Grammatik**

$P = \{$

$S \rightarrow aS \mid bA \mid cB \mid \varepsilon$

$A \rightarrow bA \mid cB \mid \varepsilon$

$B \rightarrow cB \mid \varepsilon$

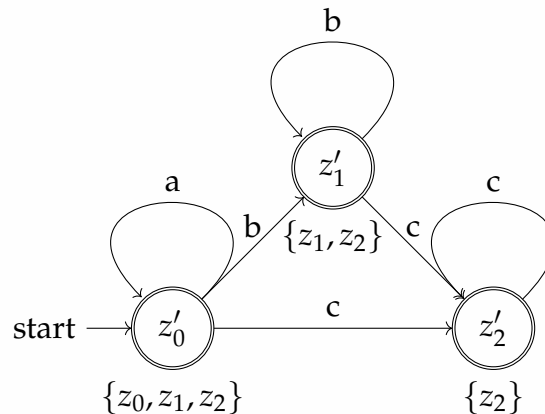
$\}$

- (b) Wandeln Sie den  $\varepsilon$ -NEA zum einem DEA mit Hilfe des erweiterter Potenzmengenalgorithmus um.

Lösungsvorschlag

Name	Zustandsmenge	Eingabe a	Eingabe b	Eingabe c
$Z_0$	$\{z_0, z_1, z_2\}$	$\{z_0, z_1, z_2\}$	$\{z_1, z_2\}$	$\{z_2\}$
$Z_1$	$\{z_1, z_2\}$	$\{\}$	$\{z_1, z_2\}$	$\{z_2\}$
$Z_2$	$\{z_2\}$	$\{\}$	$\{\}$	$\{z_2\}$

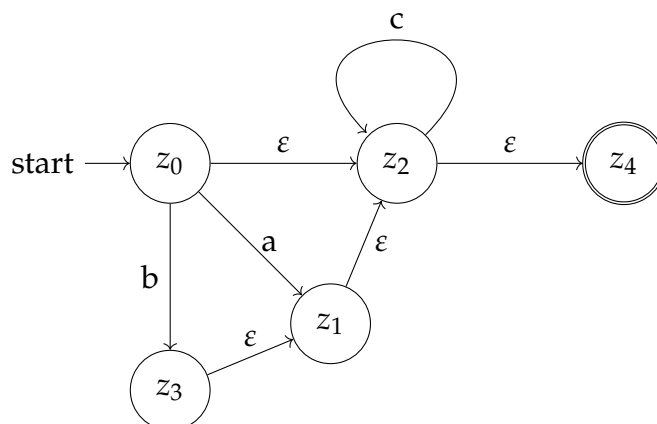
Trap-Übergänge werden aus Übersichtsgründen weg gelassen.



## Erweiterter Potenzmengenalgorithmus e-NEA zum DEA

## Erweiterter Potenzmengenalgorithmus $\varepsilon$ -NEA zum DEA

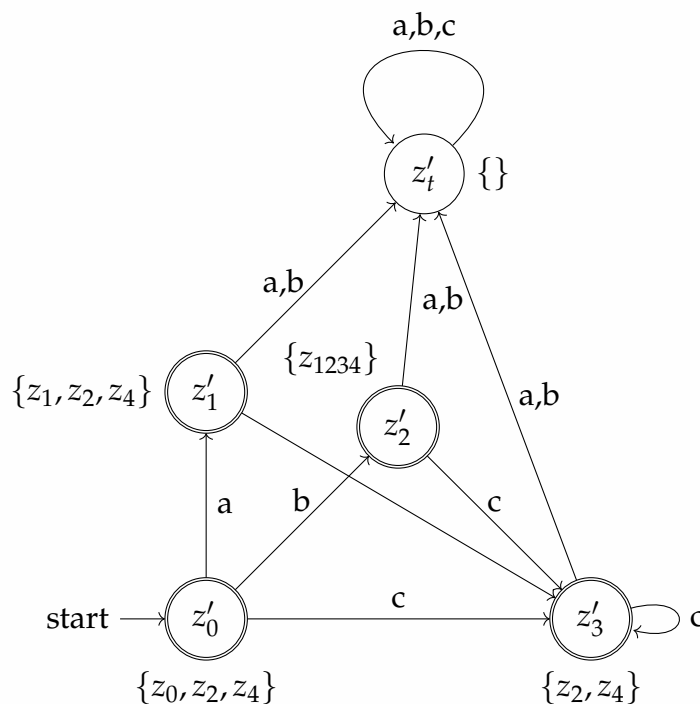
Gegeben ist der folgende  $\varepsilon$ -NEA:



Wandeln Sie den gegebenen Automaten in eine  $\varepsilon$ -freien DEA um.

Lösungsvorschlag

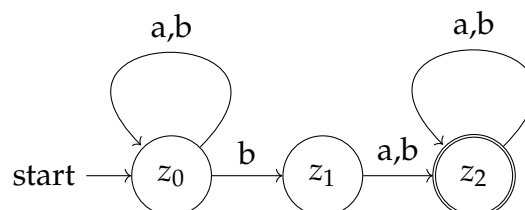
Name	Zustandsmenge	Eingabe a	Eingabe b	Eingabe c
$z'_0$	$\{z_0, z_2, z_4\}$	$\{z_1, z_2, z_4\}$	$\{z_1, z_2, z_3, z_4\}$	$\{z_2, z_4\}$
$z'_1$	$\{z_1, z_2, z_4\}$	$\{\}$	$\{\}$	$\{z_2, z_4\}$
$z'_2$	$\{z_1, z_2, z_3, z_4\}$	$\{\}$	$\{\}$	$\{z_2, z_4\}$
$z'_3$	$\{z_2, z_4\}$	$\{\}$	$\{\}$	$\{z_2, z_4\}$
$z'_t$	$\{\}$	$\{\}$	$\{\}$	$\{\}$



## Potenzmengenalgorithmus

## Potenzmengenalgorithmus

Gegeben ist der folgende NEA:

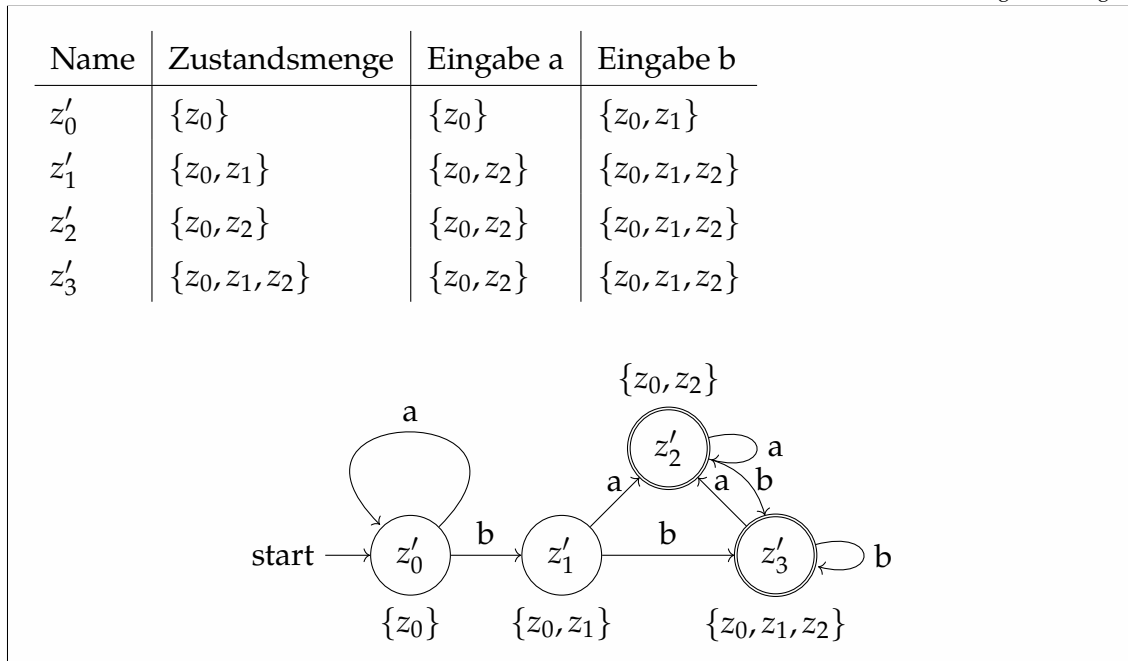


- (a) Welche Sprache akzeptiert dieser Automat? Beschreiben Sie in Worten und stellen Sie einen regulären Ausdruck hierfür auf.

$$(a|b)^*b(a|b)(a|b)^*$$

(b) Überführe den gegebenen NEA mit dem Potenzmengenalgorithmus in einen DEA.

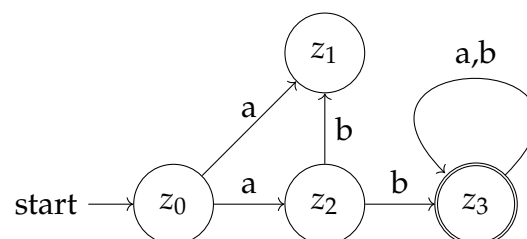
Lösungsvorschlag



## Potenzmengenalgorithmus

## Potenzmengenalgorithmus

Gegeben ist der folgende nichtdeterministische endliche Automat:

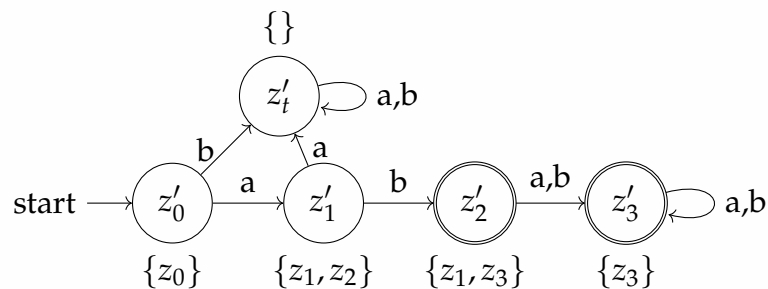


Überführen Sie den gegebenen nichtdeterministischen endlichen Automaten mit dem Potenzmengenalgorithmus in einen deterministischen endlichen Automaten.

Lösungsvorschlag



neuer Name	Zustandsmenge	Eingabe a	Eingabe b
$z'_0$	$\{z_0\}$	$\{z_1, z_2\}$	$\{\}$
$z'_1$	$\{z_1, z_2\}$	$\{\}$	$\{z_1, z_3\}$
$z'_2$	$\{z_1, z_3\}$	$\{z_3\}$	$\{z_3\}$
$z'_3$	$\{z_3\}$	$\{z_3\}$	$\{z_3\}$
$z'_t$	$\{\}$	$\{\}$	$\{\}$



## Pumping-Lemma

## Pumping-Lemma

Zeigen oder widerlegen Sie: Die folgenden Sprachen über dem Alphabet  $\Sigma = \{a, b, c\}$  sind regulär.<sup>2</sup>

### Exkurs: Pumping-Lemma für Reguläre Sprachen

Es sei  $L$  eine reguläre Sprache. Dann gibt es eine Zahl  $j$ , sodass für alle Wörter  $\omega \in L$  mit  $|\omega| \geq j$  (jedes Wort  $\omega$  in  $L$  mit Mindestlänge  $j$ ) jeweils eine Zerlegung  $\omega = uvw$  existiert, sodass die folgenden Eigenschaften erfüllt sind:

- (a)  $|v| \geq 1$  (Das Wort  $v$  ist nicht leer.)
- (b)  $|uv| \leq j$  (Die beiden Wörter  $u$  und  $v$  haben zusammen höchstens die Länge  $j$ .)
- (c) Für alle  $i = 0, 1, 2, \dots$  gilt  $uv^i w \in L$  (Für jede natürliche Zahl (mit 0)  $i$  ist das Wort  $uv^i w$  in der Sprache  $L$ )

Die kleinste Zahl  $j$ , die diese Eigenschaften erfüllt, wird Pumping-Zahl der Sprache  $L$  genannt.

$$L_1 = \{ww \mid w \in \{a, b\}^*\}$$

Lösungsvorschlag

Angenommen  $L_1$  sei regulär, dann müsste  $L_1$  die Bedingungen der stärkeren Variante des Pumping-Lemmas erfüllen.

<sup>2</sup>[https://userpages.uni-koblenz.de/~dpeuter/teaching/17ss\\_gti/blatt04\\_loesung.pdf](https://userpages.uni-koblenz.de/~dpeuter/teaching/17ss_gti/blatt04_loesung.pdf)

**Beweis durch Widerspruch:**

Sei  $j \in \mathbb{N}$  die Konstante aus dem Pumping-Lemma und  $\omega = a^j b a^j b$  ein Wort aus  $L_1$  ( $|\omega| > j$  gilt offensichtlich).

Dann müsste  $\omega$  nach dem Pumping-Lemma zerlegbar sein in  $\omega = uvw$  mit  $|v| \geq 1$  und  $|uv| < j$ .  $uv$  kann wegen  $|uv| < j$  kein  $b$  enthalten und liegt komplett im ersten  $a^j$ .

Also:

$$a^j b a^j b = uvw \text{ mit } u = a^x, v = a^y, w = a^{n-x-y} b a^j b (n \geq x + y, x > 0)$$

Dann gilt

$$uv^0 w = a^x a^{j-x-y} b a^j b = a^{j-y} b a^j b \notin L_1$$

Wir haben gezeigt, dass es keine gültige Zerlegung für  $\omega$  gibt. Also gilt für  $L_1$  die stärkere Variante des Pumping-Lemmas nicht. Somit kann  $L_1$  nicht regulär sein.

**Pumping-Lemma****Pumping-Lemma**

Begründe jeweils, ob die folgenden Sprachen regulär sind oder nicht.<sup>3</sup>

- (a)  $L_1 = \{ w \in \{a, b\}^* \mid \text{auf ein } a \text{ folgt immer ein } b \}$

Lösungsvorschlag

$L_1 = L(b^*(ab)^*b^*)$  und damit regulär.

- (b)  $L_2 = \{ w \in \{1\}^* \mid \exists n \in \mathbb{N} \text{ mit } |w| = n^2 \}$

Lösungsvorschlag

$L_2$  ist nicht regulär.

**Pumping-Lemma:**

$j$  sei eine Quadratzahl: Somit ist  $1^j \in L_2$ . Es gilt  $|uv| \leq j$  und  $|v| \geq 1$ . Daraus folgt, dass in  $v$  mindestens eine 1 existiert. Somit wird immer ein  $i \in \mathbb{N}$  existieren, sodass  $uv^i w \notin L$ , weil die Quadratzahlen nicht linear darstellbar sind.

**Begründung über die Zahlentheorie:**

Angenommen,  $L_2$  sei regulär, sei  $m$  die kleinste Zahl mit  $m^2 > j$ . Dann ist  $x = 1^{m^2} \in L_2$ . Für eine Zerlegung  $x = uvw$  nach dem Pumping-Lemma muss

<sup>3</sup><https://www.uni-muenster.de/Informatik/u/lammers//EDU/ws08/AutomatenFormaleSprachen/Loesungen/Loesung05.pdf>

dann ein  $k$  existieren mit  $v = 1^k$  und  $m^2 - l + k^l$  ist eine Quadratzahl für jedes  $l \geq 0$ . Das kann offenbar zahlentheoretisch nicht sein, und somit haben wir einen Widerspruch zur Annahme.

(c)  $L_3 = \{ a^n b^m c^n \mid m, n \in \mathbb{N}_0 \}$

Lösungsvorschlag

$L_3 = \{ a^n b^m c^n \mid m, n \in \mathbb{N}_0 \}$  ist nicht regulär.

$a^j b^j c^j \in L_3$ :

$|uv| \leq j$  und  $|v| \geq 1$

→ in  $uv$  sind nur  $a$ 's und in  $v$  ist mindestens ein  $a$

→  $uv^2w \notin L_3$ , weil dann mehr  $a$ 's als  $c$ 's in diesem Wort vorkommen

(d)  $L_4 = \{ w \in \{a\}^* \mid \text{mod } 3(|w|) = 0 \}$

Lösungsvorschlag

$L_4 = ((aaa)^*)$  und damit regulär.

## Pumping-Lemma

## Pumping-Lemma

4

### Exkurs: Pumping-Lemma für Reguläre Sprachen

Es sei  $L$  eine reguläre Sprache. Dann gibt es eine Zahl  $j$ , sodass für alle Wörter  $\omega \in L$  mit  $|\omega| \geq j$  (jedes Wort  $\omega$  in  $L$  mit Mindestlänge  $j$ ) jeweils eine Zerlegung  $\omega = uvw$  existiert, sodass die folgenden Eigenschaften erfüllt sind:

- (a)  $|v| \geq 1$  (Das Wort  $v$  ist nicht leer.)
- (b)  $|uv| \leq j$  (Die beiden Wörter  $u$  und  $v$  haben zusammen höchstens die Länge  $j$ .)
- (c) Für alle  $i = 0, 1, 2, \dots$  gilt  $uv^i w \in L$  (Für jede natürliche Zahl (mit 0)  $i$  ist das Wort  $uv^i w$  in der Sprache  $L$ )

Die kleinste Zahl  $j$ , die diese Eigenschaften erfüllt, wird Pumping-Zahl der Sprache  $L$  genannt.

$$L_1 = \{ w c w^R \mid w \in \{a, b\}^* \}$$

Erläuterung:  $w^R$  ist die Spiegelung von  $w$ , d.h. es enthält die Zeichen von  $w$  in umgekehrter Reihenfolge. Worte von  $L_1$  sind also z. B.  $c, abcba, bbbaabacabaabbb$

<sup>4</sup><http://www.coli.uni-saarland.de/courses/I2CL-10/material/Uebungsblaetter/Musterloesung4.4.pdf>

$L_1$  ist kontextfrei.

**Beweis, dass  $L_1$  nicht regulär ist, durch das Pumping Lemma:**

Wir nehmen an  $L_1$  wäre regulär. Dann gibt es einen endlichen Automaten, der  $L_1$  erkennt. Die Anzahl der Zustände dieses Automaten sei  $j$ . Wir wählen jetzt das Wort  $\omega = a^j c a^j$ .  $\omega$  liegt in  $L_1$ , und ist offensichtlich länger als  $j$ . Dieses Wort muss irgendwo eine Schleife, also einen aufpumpbaren Teil enthalten, öman kann es so in  $uvw$  zerlegen, dass für jede natürliche Zahl  $i$  auch  $uv^i w$  zu  $L_1$  gehört. Wo könnte dieser aufpumpbare Teil liegen?

**Fall 1:** Der aufpumpbare Teil  $v$  liegt komplett im Bereich des ersten  $a^j$ -Blocks. Dann würde aber  $uv^2w = a^{j+|v|} c a^j$  mehr  $a$ 's im ersten Teil als im zweiten Teil enthalten und läge nicht mehr in  $L_1$ .

**Fall 2:**  $v$  enthält das  $c$ . Dann würde aber  $uv^2w$  zwei  $c$ 's enthalten und läge damit nicht mehr in  $L_1$ .

**Fall 3:** Der aufpumpbare Teil liegt komplett im Bereich des zweiten  $a^j$ -Blocks. Dann liegt analog zu Fall 1  $uv^2w$  nicht mehr in  $L_1$ . Unser Wort lässt sich also nicht so zerlegen, dass man den Mittelteil aufpumpen kann, also ist die Annahme, dass  $L_1$  regulär ist, falsch.

Beweis, dass  $L_1$  kontextfrei ist, durch Angabe einer kontextfreien Grammatik:

$$P = \left\{ \begin{array}{l} S \rightarrow aSa \\ S \rightarrow bSb \\ S \rightarrow c \end{array} \right\}$$

## Übung zum Pumping-Lemma

## Übung zum Pumping-Lemma

### Exkurs: Pumping-Lemma für Reguläre Sprachen

Es sei  $L$  eine reguläre Sprache. Dann gibt es eine Zahl  $j$ , sodass für alle Wörter  $\omega \in L$  mit  $|\omega| \geq j$  (jedes Wort  $\omega$  in  $L$  mit Mindestlänge  $j$ ) jeweils eine Zerlegung  $\omega = uvw$  existiert, sodass die folgenden Eigenschaften erfüllt sind:

- (a)  $|v| \geq 1$  (Das Wort  $v$  ist nicht leer.)
- (b)  $|uv| \leq j$  (Die beiden Wörter  $u$  und  $v$  haben zusammen höchstens die Länge  $j$ .)

- (c) Für alle  $i = 0, 1, 2, \dots$  gilt  $uv^i w \in L$  (Für jede natürliche Zahl (mit 0)  $i$  ist das Wort  $uv^i w$  in der Sprache  $L$ )

Die kleinste Zahl  $j$ , die diese Eigenschaften erfüllt, wird Pumping-Zahl der Sprache  $L$  genannt.

- (a) Zeigen Sie, dass die Sprache  $L = \{ a^n b^m \mid n \geq m \geq 1 \}$  nicht regulär ist.

Lösungsvorschlag

$$\begin{aligned} |a^j b^j| &\geq j \\ a^j b^j &= uvw \text{ mit } |uv| \leq j \text{ und } |v| \geq 1 \\ \Rightarrow \text{in } v &\text{ nur } a\text{'s} \\ \Rightarrow uv^0 w &\notin L \end{aligned}$$

- (b) Zeige, dass die Sprache  $L = \{ a^n b^m \mid n > m \geq 1 \}$  nicht regulär ist.

Lösungsvorschlag

$$\begin{aligned} |a^{j+1} b^j| &\geq j \\ a^{j+1} b^j &= uvw \text{ mit } |uv| \leq j \text{ und } |v| \geq 1 \\ \Rightarrow \text{in } v &\text{ nur } a\text{'s} \\ \Rightarrow uv^0 w &\notin L \end{aligned}$$

## Aufgabe 4: Reguläre Ausdrücke in JAVA:

### Aufgabe 4: Reguläre Ausdrücke in JAVA: Abitur 2018 Inf2. IV. 1. und Abitur 2015 Inf2. III. 2.

- (a) Die Software einer Arztpraxis ermöglicht unter anderem die Erstellung von Rechnungen für die Patienten. Dabei muss eine Rechnungskennzahl angegeben werden, die wie folgt aufgebaut ist:

- zwei Buchstaben für die Initialen des Patienten (erster Buchstabe des Nachnamens gefolgt vom ersten Buchstaben des Vornamens)
- Bindestrich
- Patientennummer beliebiger Länge, die aus den Ziffern 0 bis 9 besteht, aber nicht mit 0 beginnt
- Versicherungsart: P bei Privatpatienten, G bei gesetzlich Versicherten
- nur bei gesetzlich Versicherten: zweistellige Versicherungskennzahl (Nummern im Bereich von 01 bis 12)
- Bindestrich
- fortlaufende Rechnungsnummer, die aus beliebig vielen Ziffern von 0 bis 9 besteht, aber nicht mit 0 beginnt

Beispiele:

Privatpatient Ingo Matik mit der Patientennummer 32 erhält seine 9. Rechnung. Die zugehörige Rechnungskennzahl ist: MI-32P-9.

Seine Frau Martha Matik mit der Patientennummer 1234, die gesetzlich bei einer Versicherung mit der Kennzahl 07 versichert ist, erhält ihre 12. Rechnung. Die zugehörige Rechnungskennzahl ist: MM-1234G07-12. Für die Darstellung der Rechnungskennzahl stehen das lateinische Alphabet der Großbuchstaben, die Ziffern 0 bis 9 und der Bindestrich zur Verfügung.

Stelle einen regulären Ausdruck für die Rechnungskennzahl in Java-Schreibweise auf.

Lösungsvorschlag

```
public class KrankenversicherungsNummer {

    public static void matches(String nummer) {
        System.out.print("Die Versicherungsnummer " + nummer + " ");
        if
→ (nummer.matches("[A-Z]{2}-[1-9]\\d*(P|(G(0[1-9]|1[0-2])))-[1-9][0-9]*"))
→ {
            System.out.print("ist eine");
        } else {
            System.out.print("ist keine");
        }
        System.out.println(" valide Nummer.");
    }

    public static void main(String[] args) {
        matches("MI-32P-9");
        matches("MM-1234G07-12");
        matches("MM-1234G17-12");
    }
}
```

Code-Beispiel auf Github ansehen:  
[src/main/java/org/bschlangaul/aufgaben/theo\\_inf/regulaere\\_ausdruecke/KrankenversicherungsNummer.java](https://github.com/bschlangaul/aufgaben/tree/main/java/org/bschlangaul/aufgaben/theo_inf/regulaere_ausdruecke/KrankenversicherungsNummer.java)

- (b) Um in die zentrale Personalabteilung der Autohauskette zu gelangen, muss man vor der Sicherheitstür ein aus genau drei Zeichen bestehendes Passwort eingeben. Dieses wird jährlich gemäß den folgenden Vorgaben der Firma festgelegt:

- Das Passwort muss mindestens einen Kleinbuchstaben und eine Ziffer enthalten;
- es darf nicht mit einem der 32 Sonderzeichen (z. B. \*, §, ...) beginnen;
- Großbuchstaben sind an keiner Stelle zugelassen.

Stelle einen regulären Ausdruck für die Rechnungskennzahl in Java-Schreibweise auf.

```

public class PasswortTuer {

    public static String b = "[a-z]";
    public static String Z = "\\d";
    public static String S =
        ↪ "[!\"#$%&'\\"\\(\\)\\*\\+,-\\.\\/;<=>\\?@\\[\\]\\^\\_`{\\|\\}\\~]";

    public static void matches(String password) {
        System.out.print("Das Passwort " + password + " ");
        // "(b(S|b)Z)|(bZ(S|Z|b))|(Zb(S|Z|b))|(Z(S|Z)b)"
        if (password.matches(String.format("%s|%s|%s|%s",
            String.format("(%s(%s|%s)%s)", b, S, b, Z),
            String.format("(%s%s(%s|%s|%s))", b, Z, S, Z, b),
            String.format("(%s%s(%s|%s|%s))", Z, b, S, Z, b),
            String.format("(%s(%s|%s)%s)", Z, S, Z, b)))) {
            System.out.print("ist ein");
        } else {
            System.out.print("ist kein");
        }
        System.out.println(" valides Passwort.");
    }

    public static void main(String[] args) {
        matches("ab1");
    }
}

```

Code-Beispiel auf Github ansehen: [src/main/java/org/bschlangaul/aufgaben/theo\\_inf/regulaere\\_ausdruecke/PasswortTuer.java](https://github.com/src/main/java/org/bschlangaul/aufgaben/theo_inf/regulaere_ausdruecke/PasswortTuer.java)

## Reguläre Grammatik, reguläre Ausdrücke und DEA

## Reguläre Grammatik, reguläre Ausdrücke und DEA

Gegeben sind die folgenden Sprachen über dem Alphabet  $\Sigma = \{a, b\}$ :

- $L_0 = \{w \mid w \text{ enthält mindestens ein } bb\}$
- $L_1 = \{w \mid w \text{ endet auf höchstens ein } b\}$
- $L_2 = \{w \mid w \text{ fängt mit } aa \text{ an oder hört mit } bb \text{ auf}\}$

(a) Geben Sie zu allen Sprachen eine reguläre Grammatik an.

Lösungsvorschlag

$G_0 = (V, \Sigma, P, S)$  mit  $V = \{S, A, B\}$ ,  $\Sigma = \{a, b\}$ ,  $S = S$  und mit

$$P = \left\{ \right.$$

$$\begin{aligned}
 S &\rightarrow aS \mid bA \\
 A &\rightarrow aS \mid bB \mid b \\
 B &\rightarrow aB \mid a \mid bB \mid b
 \end{aligned}$$

}

Der Automat auf flaci.com (FLACI: Formale Sprachen, abstrakte Automaten, Compiler und Interpreter) Ein Projekt der Hochschule Zittau/Görlitz und der Pädagogischen Hochschule Schwyz: flaci.com/Gjp92ri0w

$G_1 = (V, \Sigma, P, S)$  mit  $V = \{S, A, B\}$ ,  $\Sigma = \{a, b\}$ ,  $S = S$  und mit

$$P = \{$$

$$S \rightarrow aS \mid bS \mid b$$

}

Der Automat auf flaci.com (FLACI: Formale Sprachen, abstrakte Automaten, Compiler und Interpreter) Ein Projekt der Hochschule Zittau/Görlitz und der Pädagogischen Hochschule Schwyz: flaci.com/Gfdn0xhwg

$G_2 = (V, \Sigma, P, S)$  mit  $V = \{S, A, B, C, D, E\}$ ,  $\Sigma = \{a, b\}$ ,  $S = S$  und mit

$$P = \{$$

$$\begin{aligned}
 S &\rightarrow aA \mid bC \\
 A &\rightarrow aB \mid a \mid bC \\
 B &\rightarrow aB \mid a \mid bB \mid b \\
 C &\rightarrow aD \mid bE \mid b \\
 D &\rightarrow bC \mid aD \\
 E &\rightarrow bE \mid b \mid aD
 \end{aligned}$$

}

Der Automat auf flaci.com (FLACI: Formale Sprachen, abstrakte Automaten, Compiler und Interpreter) Ein Projekt der Hochschule Zittau/Görlitz und der Pädagogischen Hochschule Schwyz: flaci.com/Gib1z1cwi

(b) Geben Sie zu den folgenden Wörtern eine Ableitung bzw. einen Syntaxbaum anhand der erstellten Grammatiken aus der Teilaufgabe a) an:

(i) zum Wort *abba* aus der Sprache  $L_0$ .

Lösungsvorschlag

$$S \vdash aS \vdash abA \vdash abbB \vdash aabb$$

(ii) zum Wort *baab* aus der Sprache  $L_1$ .

Lösungsvorschlag

$$S \vdash bS \vdash baS \vdash baaS \vdash baab$$

(iii) zum Wort *aabb* aus der Sprache  $L_2$ .



$$S \vdash aA \vdash aaB \vdash aabB \vdash aabb$$

(c) Geben Sie zu allen Sprachen einen regulären Ausdruck an.

Lösungsvorschlag

$$\text{Reg}_0 = (a|b)^*bb(a|b)^*$$

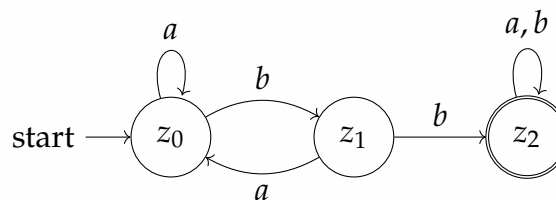
$$\text{Reg}_1 = (b|a)^*b$$

$$\text{Reg}_2 = (aa(a|b)^*)|((a|b)^*bb)$$

(d) Geben Sie zu allen Sprachen einen Automaten an, der die Sprache akzeptiert.

**Automat zu  $L_0$  :**

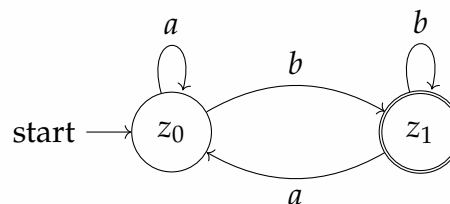
Lösungsvorschlag



Der Automat auf flaci.com (FLACI: Formale Sprachen, abstrakte Automaten, Compiler und Interpreter) Ein Projekt der Hochschule Zittau/Görlitz und der Pädagogischen Hochschule Schwyz: [flaci.com/Af75ihbc7](http://flaci.com/Af75ihbc7)

**Automat zu  $L_1$  :**

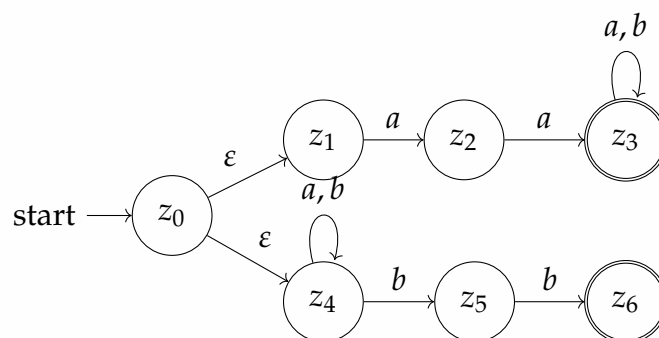
Lösungsvorschlag



Der Automat auf flaci.com (FLACI: Formale Sprachen, abstrakte Automaten, Compiler und Interpreter) Ein Projekt der Hochschule Zittau/Görlitz und der Pädagogischen Hochschule Schwyz: [flaci.com/A53w3wec9](http://flaci.com/A53w3wec9)

**Automat (NEA mit  $\varepsilon$ -Übergängen) zu  $L_2$  :**

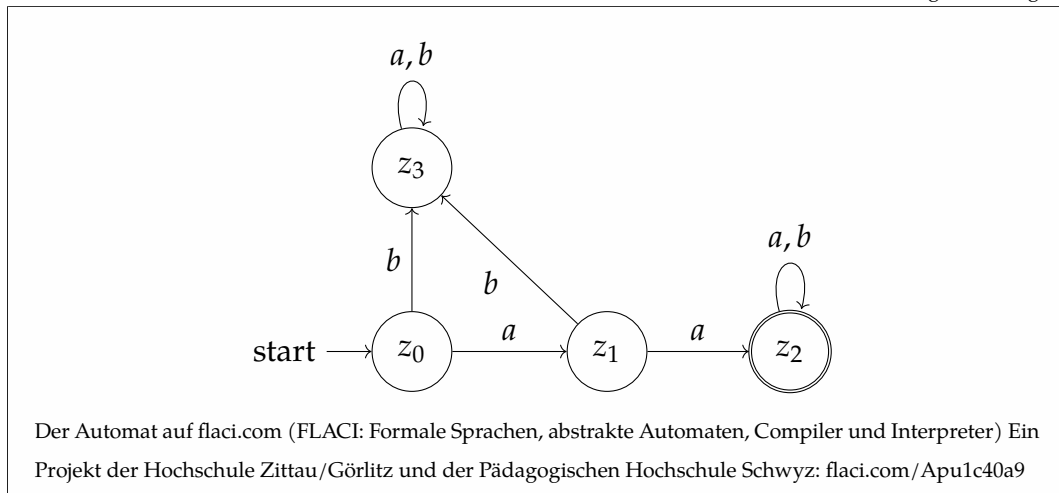
Lösungsvorschlag



Der Automat auf flaci.com (FLACI: Formale Sprachen, abstrakte Automaten, Compiler und Interpreter) Ein Projekt der Hochschule Zittau/Görlitz und der Pädagogischen Hochschule Schwyz: flaci.com/Aj5awmjba

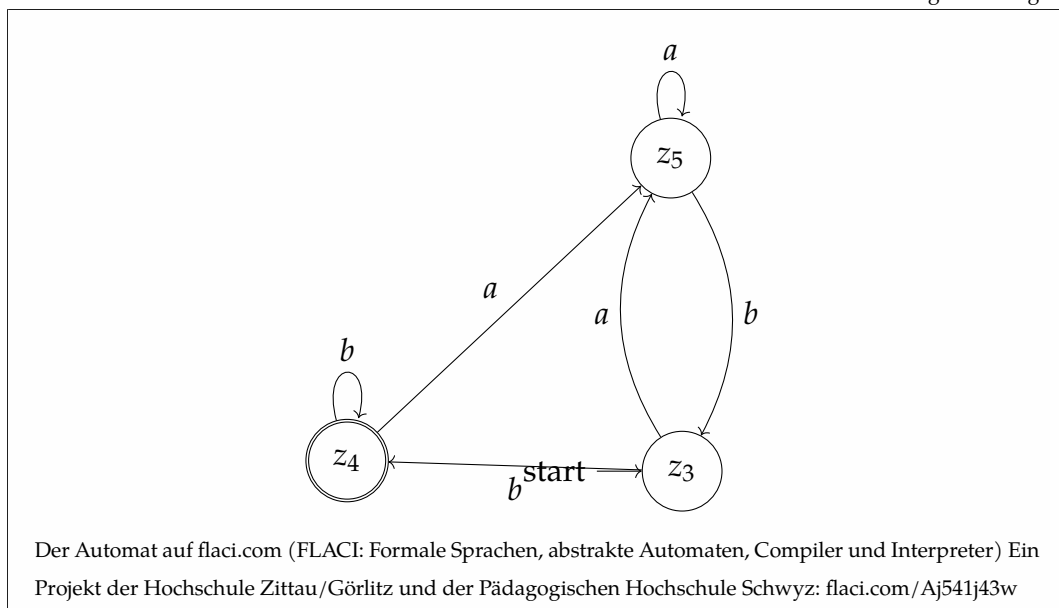
### Teil-Automat (DEA Wort beginnt mit zwei a) zu $L_2$ :

Lösungsvorschlag



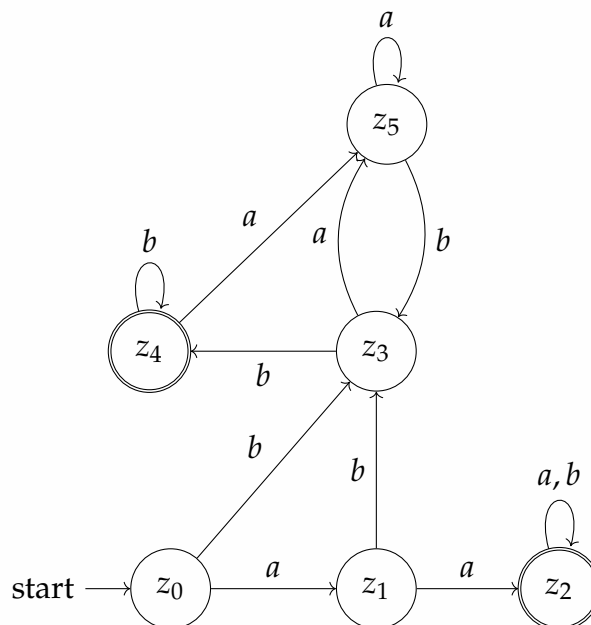
### Teil-Automat (DEA Wort endet auf zwei b) zu $L_2$ :

Lösungsvorschlag



### Automat (DEA) zu $L_2$ :

Lösungsvorschlag



Der Automat auf flaci.com (FLACI: Formale Sprachen, abstrakte Automaten, Compiler und Interpreter) Ein Projekt der Hochschule Zittau/Görlitz und der Pädagogischen Hochschule Schwyz: [flaci.com/A5ocw5ac2](http://flaci.com/A5ocw5ac2)

## Übungen zu regulären Ausdrücken

## Übungen zu regulären Ausdrücken

- (a) Gegeben ist eine Sprache  $L \subset \Sigma^*$  mit  $\Sigma = \{a, b\}$ . Zu der Sprache  $L$  gehören alle Wörter, die die Zeichenfolge abba beinhalten.

Geben Sie einen regulären Ausdruck für diese Sprache an („klassischer“ regulärer Ausdruck).

Lösungsvorschlag

$$(a|b)^*abba(a|b)^*$$

```
static String regexABBA = "(a|b)*abba(a|b)*";
```

Code-Beispiel auf Github ansehen: [src/main/java/org/bschlangaul/aufgaben/theo\\_inf/regulaere\\_ausdruecke/TestRegularExpressions.java](https://github.com/bschlangaul/aufgaben/theo_inf/regulaere_ausdruecke/TestRegularExpressions.java)

- (b) Gebe möglichst einfache reguläre Ausdrücke für die folgenden Sprachen  $L_x \subset \Sigma^*$  mit  $\Sigma = \{a, b\}$  und  $x \in \{1, 2, 3\}$  („klassischer“ regulärer Ausdruck).

$$L_1 = \{x | x \text{ beinhaltet eine gerade Anzahl von } a\}$$

$b^*(ab^*ab^*)^*$

```
static String regexGeradeA = "b*(ab*ab*)*"; // Epsilon aa aaaa abba  
↪ bbaa aabb bbb
```

[Code-Beispiel auf Github ansehen:](#)

```
src/main/java/org/bschlangaul/aufgaben/theo_inf/regulaere_ausdruecke/TestRegularExpressions.java
```

$$L_2 = \{x \mid x \text{ beinhaltet eine ungerade Anzahl von } b\}$$

Lösungsvorschlag

$$a^*ba^*(ba^*ba^*)^*$$

```
static String regexUngeradeB = "a*ba*(ba*ba*)*"; // Epsilon b bbb
↪ abababa
```

Code-Beispiel auf Github ansehen:  
src/main/java/org/bschlangaul/aufgaben/theo\_inf/regulaere\_ausdruecke/TestRegularExpressions.java

$$L_3 = \{x \mid x \text{ beinhaltet an seinen geradzahlgigen Positionen ausschließlich } a\}$$

Lösungsvorschlag

$$((a|b)a)^*(a^*|b)$$

```
static String regexGeradzahligA = "((a|b)a)*(a*|b)"; // aa ba aab b
```

Code-Beispiel auf Github ansehen:  
src/main/java/org/bschlangaul/aufgaben/theo\_inf/regulaere\_ausdruecke/TestRegularExpressions.java

- (c) Geben Sie einen regulären Ausdruck an, der eine syntaktisch gültige E-Mail-Adresse erkennt. (mindestens 1 Zeichen (Groß-/Kleinbuchstabe oder Zahl) vor dem @; mindestens 1 Zeichen (Groß-/Kleinbuchstabe oder Zahl) nach dem @; alle E-Mail-Adressen sollen auf .de oder .com enden.

Lösungsvorschlag

```
static String regexEMAIL = "\\w+@\\w+\\. (de|com)";
```

Code-Beispiel auf Github ansehen: src/main/java/org/bschlangaul/aufgaben/theo\_inf/regulaere\_ausdruecke/TestRegularExpressions.java

Lösungsvorschlag

```
public class TestRegularExpressions {
    // Hier bitte Lösungen der Aufgaben eintragen.
    static String regexABBA = "(a|b)*abba(a|b)*";
    static String regexGeradeA = "b*(ab*ab*)*"; // Epsilon aa aaaa abba bbaa aabb
    ↪ bbb
    static String regexUngeradeB = "a*ba*(ba*ba*)*"; // Epsilon b bbb abababa
    static String regexGeradzahligA = "((a|b)a)*(a*|b)"; // aa ba aab b
    static String regexEMAIL = "\\w+@\\w+\\. (de|com)";
    // Wenn die Lösungen stimmen, geben alle Tests true aus

    // Alternativen:
    // static String regexGeradzahligA = "((a|b)a)*((a|b)|)"; // aa ba aab b

    public static void main(String[] args) {
        testregexABBA();
        testregexGeradeA();
        testregexUngeradeB();
        testregexGeradzahligA();
        testregexEMAIL();
    }
}
```

```

public static void testregexABBA() {
    boolean[] b = new boolean[7];
    b[0] = "abba".matches(regexABBA);
    b[1] = !"aba".matches(regexABBA);
    b[2] = "abbaabbaabba".matches(regexABBA);
    b[3] = "abababbaaabaabaaba".matches(regexABBA);
    b[4] = !"ab".matches(regexABBA);
    b[5] = !"bbb".matches(regexABBA);
    b[6] = !"".matches(regexABBA);
    if (b[0] & b[1] & b[2] & b[3] & b[4] & b[5] & b[6]) {
        System.out.println("Alle ABBA-Tests bestanden. Dein RegEx stimmt!");
    } else {
        for (int i = 0; i < b.length; i++) {
            if (!b[i]) {
                System.out.println("Test mit dem Index" + i +
→ " leider nicht bestanden.");
            }
        }
    }
}

public static void testregexGeradeA() {
    boolean[] b = new boolean[7];
    b[0] = "aa".matches(regexGeradeA);
    b[1] = !"aaa".matches(regexGeradeA);
    b[2] = "abbaabbaabba".matches(regexGeradeA);
    b[3] = !"abababbaaabaabaaba".matches(regexGeradeA);
    b[4] = !"ab".matches(regexGeradeA);
    b[5] = "bbb".matches(regexGeradeA);
    b[6] = "".matches(regexGeradeA);
    if (b[0] & b[1] & b[2] & b[3] & b[4] & b[5] & b[6]) {
        System.out.println("Alle GeradeA-Tests bestanden. Dein RegEx stimmt!");
    } else {
        for (int i = 0; i < b.length; i++) {
            if (!b[i]) {
                System.out.println("Test mit dem Index" + i +
→ " leider nicht bestanden.");
            }
        }
    }
}

public static void testregexUngeradeB() {
    boolean[] b = new boolean[7];
    b[0] = "b".matches(regexUngeradeB);
    b[1] = !"bb".matches(regexUngeradeB);
    b[2] = !"abbaabbaabba".matches(regexUngeradeB);
    b[3] = "abababbaaabaabaaba".matches(regexUngeradeB);
    b[4] = "ab".matches(regexUngeradeB);
    b[5] = "bbb".matches(regexUngeradeB);
    b[6] = !"".matches(regexUngeradeB);
    if (b[0] & b[1] & b[2] & b[3] & b[4] & b[5] & b[6]) {
        System.out.println("Alle UngeradeB-Tests bestanden. Dein RegEx stimmt!");
    } else {

```

```

        for (int i = 0; i < b.length; i++) {
            if (!b[i]) {
                System.out.println("Test mit dem Index" + i +
→ " leider nicht bestanden.");
            }
        }
    }
}

public static void testregexGeradzahligA() {
    boolean[] b = new boolean[7];
    b[0] = !"ab".matches(regexGeradzahligA);
    b[1] = "b".matches(regexGeradzahligA);
    b[2] = "babab".matches(regexGeradzahligA);
    b[3] = !"bababaab".matches(regexGeradzahligA);
    b[4] = "ba".matches(regexGeradzahligA);
    b[5] = "aaa".matches(regexGeradzahligA);
    b[6] = "".matches(regexGeradzahligA);
    if (b[0] & b[1] & b[2] & b[3] & b[4] & b[5] & b[6]) {
→ System.out.println("Alle GeradzahligeA-Tests bestanden. Dein RegEx stimmt!");
    } else {
        for (int i = 0; i < b.length; i++) {
            if (!b[i]) {
                System.out.println("Test mit dem Index" + i +
→ " leider nicht bestanden.");
            }
        }
    }
}

public static void testregexEMAIL() {
    boolean[] b = new boolean[7];
    b[0] = "3@s.de".matches(regexEMAIL);
    b[1] = !"@0.de".matches(regexEMAIL);
    b[2] = "asdf@asdf.com".matches(regexEMAIL);
    b[3] = !"@.de".matches(regexEMAIL);
    b[4] = "s@1.com".matches(regexEMAIL);
    b[5] = !"a@a".matches(regexEMAIL);
    b[6] = !"".matches(regexEMAIL);
    if (b[0] & b[1] & b[2] & b[3] & b[4] & b[5] & b[6]) {
        System.out.println("Alle Email-Tests bestanden. Dein RegEx stimmt!");
    } else {
        for (int i = 0; i < b.length; i++) {
            if (!b[i]) {
                System.out.println("Test mit dem Index" + i +
→ " leider nicht bestanden.");
            }
        }
    }
}
}

```

Code-Beispiel auf Github ansehen: [src/main/java/org/bschlangaul/aufgaben/theo\\_inf/regulaere\\_ausdruecke/TestRegularExpressions.java](https://github.com/bschlangaul/aufgaben/theo_inf/regulaere_ausdruecke/TestRegularExpressions.java)

## Reguläre Sprache in kontextfreier Sprache

## Reguläre Sprache in kontextfreier Sprache

Zeigen Sie, dass sich eine reguläre Sprache ebenfalls als kontextfreie Sprache auffassen lässt.

Lösungsvorschlag

Im Grunde genommen kann ein DPDA einen deterministischen endlichen Automaten simulieren. Weil ein PDA einen Stack besitzen muss, erhält der PDA ein Symbol  $Z_0$  auf seinem Stack. Der PDA ignoriert den Stack aber und arbeitet lediglich mit seinen Zuständen. Formal ausgedrückt, sei

$$A = (Q, \Sigma, \delta_A, q_0, F)$$

ein DFA. Wir konstruieren einen DPDA

$$P = (Q, \Sigma, \{Z_0\}, \delta_P, q_0, Z_0, F),$$

indem  $\delta_P(q, a, Z_0) = \{(p, Z_0)\}$  für alle Zustände  $p$  und  $q$  aus  $Q$  definiert wird, derart dass  $\delta_A(q, a) = p$ . Wir behaupten, dass  $(q_0, w, Z_0) \rightarrow (p, \varepsilon, Z_0)$  genau dann, wenn  $\delta_A(q_0, w) = p$ . Das heißt,  $P$  simuliert  $A$  über seinen Zustand. Beide Richtungen lassen sich durch einfache Induktionsbeweise über  $|w|$  zeigen. Da sowohl  $A$  als auch  $P$  akzeptieren, indem sie einen der Zustände aus  $F$  annehmen, schließen wir darauf, dass ihre Sprachen identisch sind.

### 46115 / 2010 / Frühjahr / Thema 2 / Aufgabe 1

## Aufgabe 1

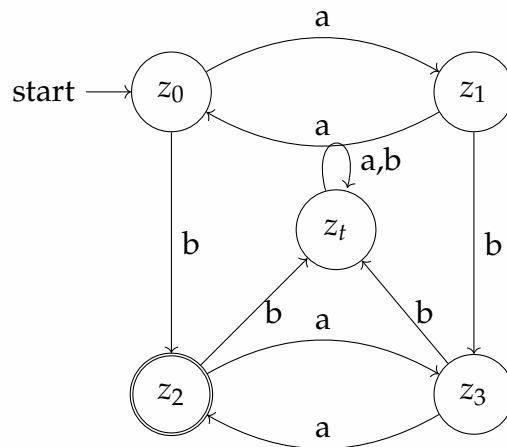
(a) Gegeben ist die folgende Sprache  $L1$  über dem Alphabet  $\Sigma = \{a, b\}$ :

$L1 = \{w \in \Sigma^* \mid \text{die Anzahl der } a \text{ in } w \text{ ist gerade und } b \text{ kommt in } w \text{ genau einmal vor}\}.$

(i) Geben Sie einen deterministischen endlichen Automaten an, der die Sprache  $L1$  akzeptiert.

Lösungsvorschlag





Der Automat auf flaci.com (FLACI: Formale Sprachen, abstrakte Automaten, Compiler und Interpreter) Ein Projekt der Hochschule Zittau/Görlitz und der Pädagogischen Hochschule Schwyz: flaci.com/Ahfqe3eb7

(ii) Geben Sie einen regulären Ausdruck an, der die Sprache  $L1$  beschreibt.

Lösungsvorschlag

$(aa)^*(b|aba)(aa)^*$

(b) Die folgende Sprache  $L2$  ist eine Erweiterung von  $L1$ :

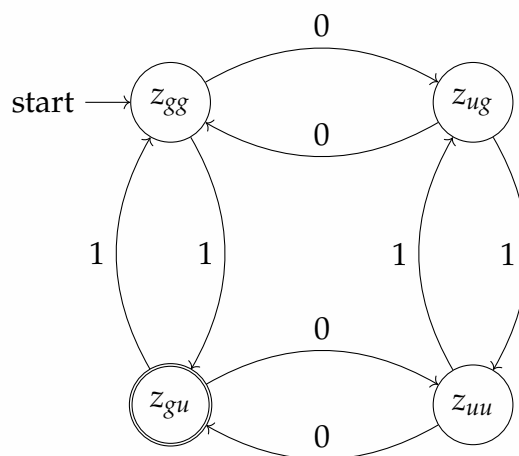
$L1 = \{w \in \Sigma^* \mid \text{die Anzahl der } a \text{ in } w \text{ ist gerade und die Anzahl der } b \text{ in } w \text{ ist ungerade}\}.$

(i) Geben Sie einen deterministischen endlichen Automaten an, der die Sprache  $L2$  akzeptiert.

Lösungsvorschlag

$a$

gu = gerade Anzahl a's, ungerade Anzahl b's  
ug = ungerade Anzahl a's, gerade Anzahl b's



Der Automat auf flaci.com (FLACI: Formale Sprachen, abstrakte Automaten, Compiler und Interpreter) Ein

Projekt der Hochschule Zittau/Görlitz und der Pädagogischen Hochschule Schwyz: [flaci.com/Af0vcjys9](https://flaci.com/Af0vcjys9)

<sup>a</sup>[https://www.informatik.uni-hamburg.de/TGI/lehre/v1/SS14/FGI1/Folien/fgi1\\_v2\\_handout.pdf](https://www.informatik.uni-hamburg.de/TGI/lehre/v1/SS14/FGI1/Folien/fgi1_v2_handout.pdf)

Reguläre Sprache

(ii) Geben Sie eine rechtslineare Grammatik an, die die Sprache  $L_2$  erzeugt.

Lösungsvorschlag

$$P = \left\{ \begin{array}{l} A \rightarrow aB \mid bD \mid b \\ B \rightarrow bC \mid aA \\ C \rightarrow aD \mid a \mid bB \\ D \rightarrow bA \mid aC \end{array} \right\}$$

<https://flaci.com/Ahfqe3eb7>

## 46115 / 2015 / Frühjahr / Thema 1 / Aufgabe 1

### Aufgabe 1

Gegeben sei die Sprache  $L$ .  $L$  besteht aus der Menge aller Worte über dem Alphabet  $\{a, b, c\}$ , die mit  $a$  beginnen und mit  $b$  enden und die nie zwei aufeinander folgende  $c$ 's enthalten.

(a) Geben Sie einen regulären Ausdruck für  $L$  an.

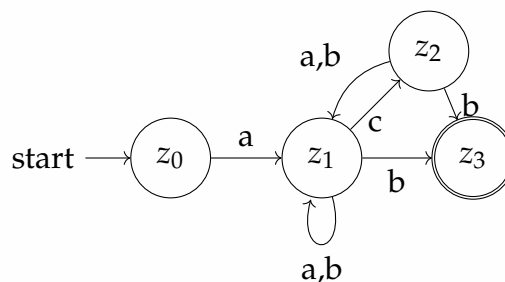
Lösungsvorschlag

$a(c?[ab]^+)*(cb|b)$

$a(c(a|b)|(a|b))*(cb|b)$

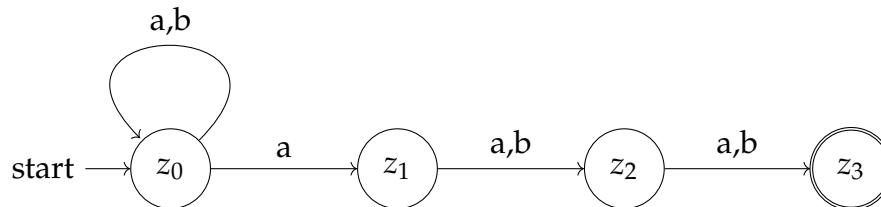
(b) Geben Sie einen vollständigen deterministischen endlichen Automaten für  $L$  an.

Lösungsvorschlag



## 46115 / 2016 / Herbst / Thema 1 / Aufgabe 1

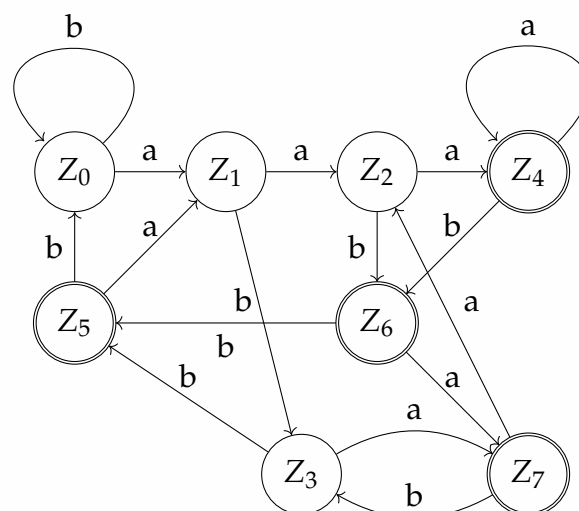
- (a) Konstruieren Sie aus dem NEA mit der Potenzmengenkonstruktion einen (deterministischen) EA, der dieselbe Sprache akzeptiert.



Der Automat auf flaci.com (FLACI: Formale Sprachen, abstrakte Automaten, Compiler und Interpreter) Ein Projekt der Hochschule Zittau/Görlitz und der Pädagogischen Hochschule Schwyz: [flaci.com/Aiqxdazuw](http://flaci.com/Aiqxdazuw)

Lösungsvorschlag

Name	Zustandsmenge	Eingabe a	Eingabe b
$Z_0$	$Z_0\{z_0\}$	$Z_1\{z_0, z_1\}$	$Z_0\{z_0\}$
$Z_1$	$Z_1\{z_0, z_1\}$	$Z_2\{z_0, z_1, z_2\}$	$Z_3\{z_0, z_2\}$
$Z_2$	$Z_2\{z_0, z_1, z_2\}$	$Z_4\{z_0, z_1, z_2, z_3\}$	$Z_6\{z_0, z_2, z_3\}$
$Z_3$	$Z_3\{z_0, z_2\}$	$Z_7\{z_0, z_1, z_3\}$	$Z_5\{z_0, z_3\}$
$Z_4$	$Z_4\{z_0, z_1, z_2, z_3\}$	$Z_4\{z_0, z_1, z_2, z_3\}$	$Z_6\{z_0, z_2, z_3\}$
$Z_5$	$Z_5\{z_0, z_3\}$	$Z_1\{z_0, z_1\}$	$Z_0\{z_0\}$
$Z_6$	$Z_6\{z_0, z_2, z_3\}$	$Z_7\{z_0, z_1, z_3\}$	$Z_5\{z_0, z_3\}$
$Z_7$	$Z_7\{z_0, z_1, z_3\}$	$Z_2\{z_0, z_1, z_2\}$	$Z_3\{z_0, z_2\}$



Der Automat auf flaci.com (FLACI: Formale Sprachen, abstrakte Automaten, Compiler und Interpreter) Ein Projekt der Hochschule Zittau/Görlitz und der Pädagogischen Hochschule Schwyz: [flaci.com/Aiqnk06c9](http://flaci.com/Aiqnk06c9)

- (b) Beschreiben Sie möglichst einfach, welche Sprachen von den folgenden regulären Ausdrücken beschrieben werden:

(i)  $(a|b)^*a$

Lösungsvorschlag

Sprache mit dem Alphabet  $\Sigma = \{a, b\}$ : Alle Wörter der Sprache enden auf  $a$ .

(ii)  $(a|b)^*a(a|b)^*a(a|b)^*$

Lösungsvorschlag

Sprache mit dem Alphabet  $\Sigma = \{a, b\}$ : Alle Wörter der Sprache enthalten mindestens 2  $a$ 's.

(iii)  $(a|b)^*a(bb)^*a(a|b)^*$

Lösungsvorschlag

Sprache mit dem Alphabet  $\Sigma = \{a, b\}$ : Alle Wörter der Sprache enthalten mindestens 2  $a$ 's, die von einer geradzahligen Anzahl von  $b$ 's getrennt sind.

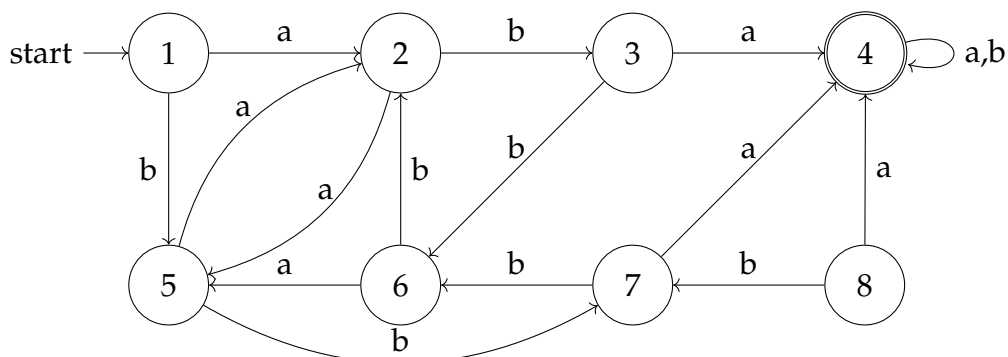
## 46115 / 2019 / Herbst / Thema 1 / Aufgabe 1

- (a) Geben Sie einen regulären Ausdruck für die Sprache über dem Alphabet  $\{a, b\}$  an, die genau alle Wörter enthält, die eine gerade Anzahl  $a$ 's haben.

Lösungsvorschlag

$$b^*(ab^*ab^*)^*$$

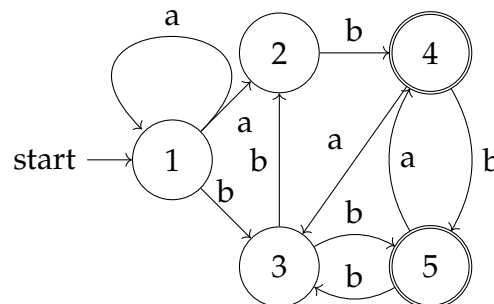
- (b) Sei  $A$  der folgende DEA über dem Alphabet  $\{a, b\}$ :



Führen Sie den Minimierungsalgorithmus für  $A$  durch und geben Sie den minimalen äquivalenten DEA für  $L(A)$  als Zeichnung an.

**46115 / 2019 / Herbst / Thema 2 / Aufgabe 1**

Es sei der nichtdeterministische endliche Automat  $A = (\{1, 2, 3, 4, 5\}, \{a, b\}, \delta, \{4, 5\}, 1)$  gegeben, wobei  $\delta$  durch folgenden Zeichnung beschrieben ist.

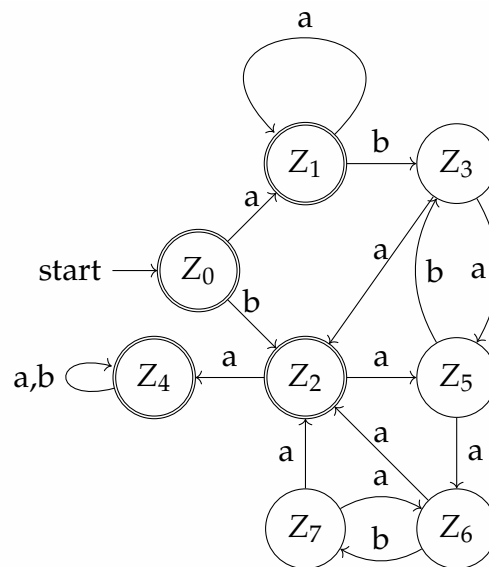


Konstruieren Sie nachvollziehbar einen deterministischen endlichen Automaten  $A'$ , der das Komplement von  $L(A)$  akzeptiert!

Lösungsvorschlag

Zuerst mit Hilfe der Potenzmengenkonstruktion einen deterministischen endlichen Automaten erstellen und dann die Zustände mit den Endzuständen tauschen.

Name	Zustandsmenge	Eingabe $a$	Eingabe $b$
$Z_0$	$Z_0\{1\}$	$Z_1\{1,2\}$	$Z_2\{3\}$
$Z_1$	$Z_1\{1,2\}$	$Z_1\{1,2\}$	$Z_3\{3,4\}$
$Z_2$	$Z_2\{3\}$	$Z_4\{\}$	$Z_5\{2,5\}$
$Z_3$	$Z_3\{3,4\}$	$Z_2\{3\}$	$Z_5\{2,5\}$
$Z_4$	$Z_4\{\}$	$Z_4\{\}$	$Z_4\{\}$
$Z_5$	$Z_5\{2,5\}$	$Z_6\{4\}$	$Z_3\{3,4\}$
$Z_6$	$Z_6\{4\}$	$Z_2\{3\}$	$Z_7\{5\}$
$Z_7$	$Z_7\{5\}$	$Z_6\{4\}$	$Z_2\{3\}$



## 46115 / 2019 / Herbst / Thema 2 / Aufgabe 2

Gegeben ist die rechtslineare Grammatik  $G = (\{a, b\}, \{S, A, B, C, D\}, S, P)$  mit

$P = \{$

$S \rightarrow aA$

$A \rightarrow bB$

$A \rightarrow aD$

$B \rightarrow aC$

$B \rightarrow bB$

$C \rightarrow bD$

$C \rightarrow b$

$D \rightarrow aA$

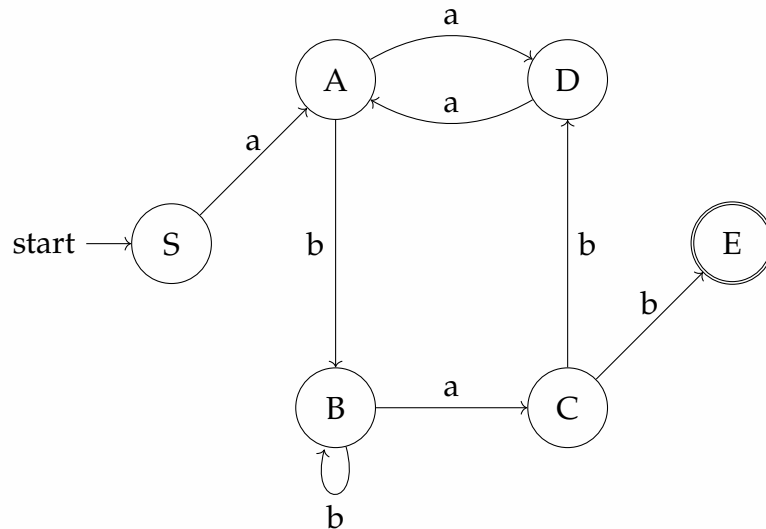
$\}$

. Sei  $L$  die von  $G$  erzeugte Sprache.

Der Automat auf [flaci.com](http://flaci.com) (FLACI: Formale Sprachen, abstrakte Automaten, Compiler und Interpreter) Ein Projekt der Hochschule Zittau/Görlitz und der Pädagogischen Hochschule Schwyz: [flaci.com/Gpkv4ansc](http://flaci.com/Gpkv4ansc)

- (a) Zeichnen Sie einen nichtdeterministischen endlichen Automaten, der  $L$  akzeptiert!

Lösungsvorschlag



Der Automat auf flaci.com (FLACI: Formale Sprachen, abstrakte Automaten, Compiler und Interpreter) Ein Projekt der Hochschule Zittau/Görlitz und der Pädagogischen Hochschule Schwyz: [flaci.com/Ahh979eqz](http://flaci.com/Ahh979eqz)

- (b) Konstruieren Sie auf nachvollziehbare Weise einen regulären Ausdruck  $\alpha$  mit  $L(\alpha) = L$ !

Lösungsvorschlag

$(ab + ab|(a|b+) + ab + ab)$  (von Flaci automatisch konvertiert)

## 46115 / 2020 / Frühjahr / Thema 1 / Aufgabe 1

- (a) Betrachten Sie die formale Sprache  $L \subseteq \{0,1\}^*$  aller Wörter, die 01 oder 110 als Teilwort enthalten.

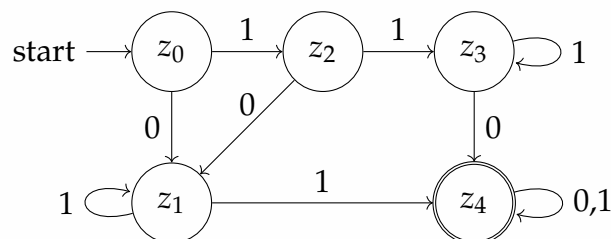
Geben Sie einen regulären Ausdruck für die Sprache  $L$  an.

Lösungsvorschlag

$(0|1)^*(01|110)(0|1)^*$

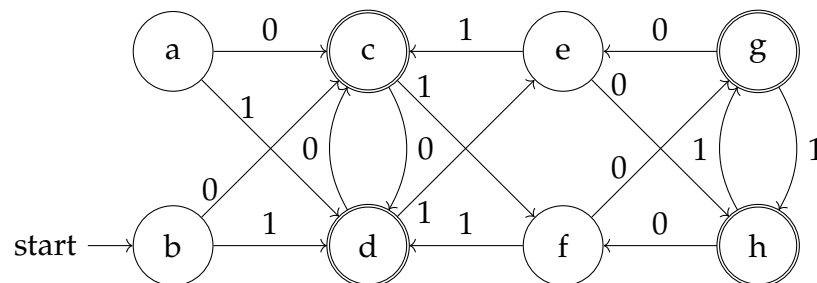
- (b) Entwerfen Sie einen (vollständigen) deterministischen endlichen Automaten, der die Sprache  $L$  aus Teilaufgabe (a) akzeptiert. (Hinweis: es werden nicht mehr als 6 Zustände benötigt.)

Lösungsvorschlag



Der Automat auf flaci.com (FLACI: Formale Sprachen, abstrakte Automaten, Compiler und Interpreter) Ein Projekt

- (c) Minimieren Sie den folgenden deterministischen endlichen Automaten:  
Machen Sie dabei Ihren Rechenweg deutlich!



Der Automat auf flaci.com (FLACI: Formale Sprachen, abstrakte Automaten, Compiler und Interpreter) Ein Projekt der Hochschule Zittau/Görlitz und der Pädagogischen Hochschule Schwyz: flaci.com/Ajpw4j73w

Lösungsvorschlag

a	∅	∅	∅	∅	∅	∅	∅	∅
b		∅	∅	∅	∅	∅	∅	∅
c	$x_1$	$x_1$	∅	∅	∅	∅	∅	∅
d	$x_1$	$x_1$		∅	∅	∅	∅	∅
e	$x_2$	$x_2$	$x_1$	$x_1$	∅	∅	∅	∅
f	$x_3$	$x_3$	$x_1$	$x_1$		∅	∅	∅
g	$x_1$	$x_1$	$x_2$	$x_2$	$x_1$	$x_1$	∅	∅
h	$x_1$	$x_1$	$x_2$	$x_2$	$x_1$	$x_1$		∅
	a	b	<u>c</u>	<u>d</u>	e	f	<u>g</u>	<u>h</u>

$x_1$  Paar aus End-/ Nicht-Endzustand kann nicht äquivalent sein.

$x_2$  Test, ob man mit der Eingabe zu einem bereits markiertem Paar kommt.

$x_3$  In weiteren Iterationen markierte Zustände.

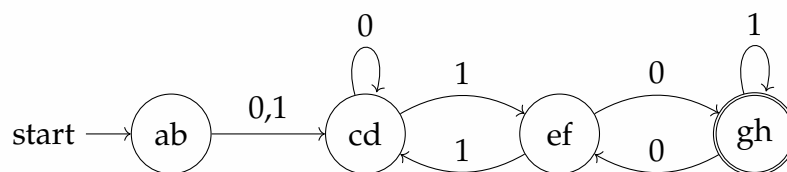
$x_4$  ...

Die Zustandpaare werden aufsteigend sortiert notiert.

**Übergangstabelle**



Zustandspaar	0	1
(a, b)	(c, c)	(d, d)
(a, e)	(c, h) $x_2$	(c, d)
(a, f)	(c, g) $x_3$	(d, d)
(b, e)	(c, h) $x_2$	(c, d)
(b, f)	(c, g) $x_3$	(d, g)
(c, d)	(c, d)	(e, f)
(c, g)	(d, e) $x_2$	(e, f)
(c, h)	(d, f) $x_2$	(f, f)
(d, g)	(c, e) $x_2$	(e, e)
(d, h)	(c, f) $x_2$	(e, f)
(e, f)	(g, h)	(c, d)
(g, h)	(e, f)	(g, h)



Der Automat auf flaci.com (FLACI: Formale Sprachen, abstrakte Automaten, Compiler und Interpreter) Ein Projekt der Hochschule Zittau/Görlitz und der Pädagogischen Hochschule Schwyz: flaci.com/Arzvh5kyz

- (d) Ist die folgende Aussage richtig oder falsch? Begründen Sie Ihre Antwort!  
 „Zu jeder regulären Sprache  $L$  über dem Alphabet  $\Sigma$  gibt es eine Sprache  $L' \subseteq \Sigma^*$ , die  $L$  enthält ( $\emptyset \subseteq L$ ) und nicht regulär ist.“

### 46115 / 2021 / Frühjahr / Thema 1 / Teilaufgabe 1 / Aufgabe 1

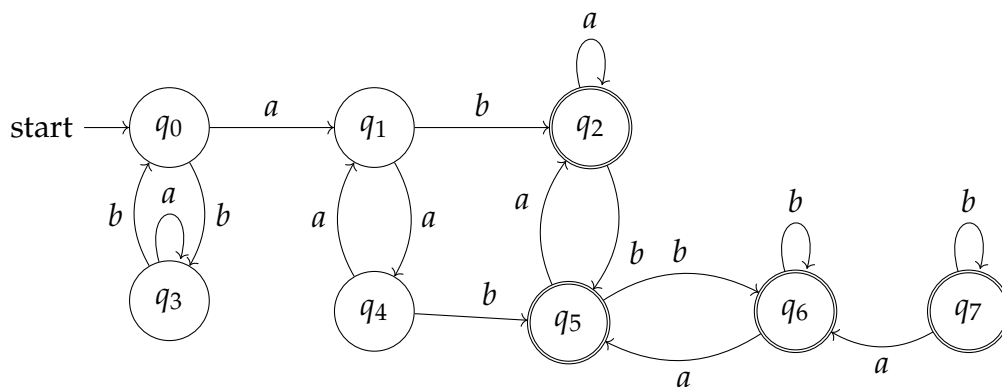
Ein moderner Kaugummi-Automat erhalte 20- und 50-Cent-Münzen. Eine Packung Kaugummi kostet 120 Cent. Die interne Steuerung des Kaugummi-Automaten verwendet einen deterministischen endlichen Automaten, der die Eingabe als Folge von 20- und 50-Cent-Münzen (d. h. als Wort über dem Alphabet 20,50) erhält, und genau die Folgen akzeptiert, die in der Summe 120 Cent ergeben.

- (a) Geben Sie zwei Worte aus  $20,50^*$  an, die der Automat akzeptiert.
- (b) Zeichnen Sie einen deterministischen endlichen Automaten als Zustandsgraph, der für die interne Steuerung des Kaugummi-Automaten verwendet werden kann (d. h. der Automat akzeptiert genau alle Folgen von 20- und 50-Cent-Münzen, deren Summe 120 ergibt).

- (c) Geben Sie einen regulären Ausdruck an, der die akzeptierte Sprache des Automaten erzeugt.

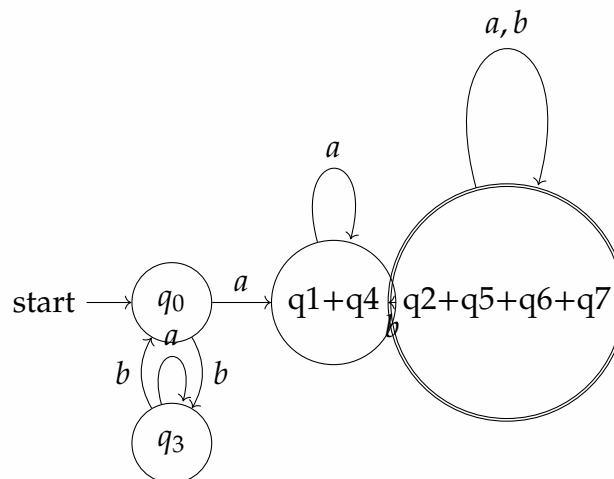
### 46115 / 2021 / Frühjahr / Thema 1 / Teilaufgabe 1 / Aufgabe 3

Betrachten Sie den unten gezeigten deterministischen endlichen Automaten, der Worte über dem Alphabet  $X = a, b$  verarbeitet. Bestimmen Sie den dazugehörigen Minimalautomaten, d. h. einen deterministischen endlichen Automaten, der die gleiche Sprache akzeptiert und eine minimale Anzahl an Zuständen benutzt. Erläutern Sie Ihre Berechnung, indem Sie z. B. eine Minimierungstabelle angeben.



Der Automat auf flaci.com (FLACI: Formale Sprachen, abstrakte Automaten, Compiler und Interpreter) Ein Projekt der Hochschule Zittau/Görlitz und der Pädagogischen Hochschule Schwyz: flaci.com/Ah5v10or9

Lösungsvorschlag



Der Automat auf flaci.com (FLACI: Formale Sprachen, abstrakte Automaten, Compiler und Interpreter) Ein Projekt der Hochschule Zittau/Görlitz und der Pädagogischen Hochschule Schwyz: flaci.com/Apkyuoo1g

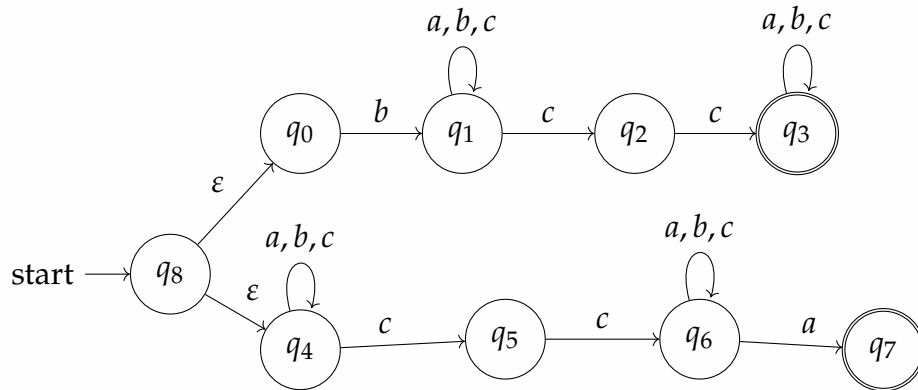
### 46115 / 2021 / Frühjahr / Thema 2 / Teilaufgabe 1 / Aufgabe 1

- (a) Betrachten Sie die formale Sprache  $L = \{a, b, c^*\}$ : aller Wörter, die entweder mit  $b$  beginnen oder mit  $a$  enden (aber nicht beides gleichzeitig) und das Teilwort  $cc$

enthalten. Entwerfen Sie einen (vollständigen) deterministischen endlichen Automaten, der die Sprache  $L$  akzeptiert. (Hinweis: Es werden weniger als 10 Zustände benötigt.)

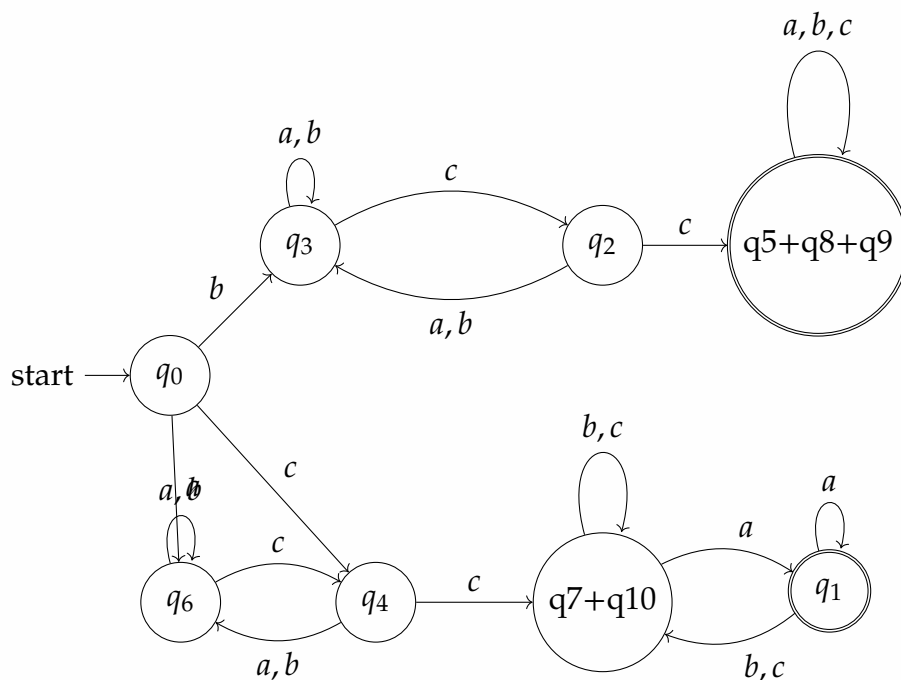
Lösungsvorschlag

NEA:



Der Automat auf flaci.com (FLACI: Formale Sprachen, abstrakte Automaten, Compiler und Interpreter) Ein Projekt der Hochschule Zittau/Görlitz und der Pädagogischen Hochschule Schwyz: [flaci.com/Ar3pvv7ha](http://flaci.com/Ar3pvv7ha)

konvertierter DEA:



Der Automat auf flaci.com (FLACI: Formale Sprachen, abstrakte Automaten, Compiler und Interpreter) Ein Projekt der Hochschule Zittau/Görlitz und der Pädagogischen Hochschule Schwyz: [flaci.com/Ai89m0txw](http://flaci.com/Ai89m0txw)

(b) Ist die folgende Aussage richtig? Begründen Sie Ihre Antwort.

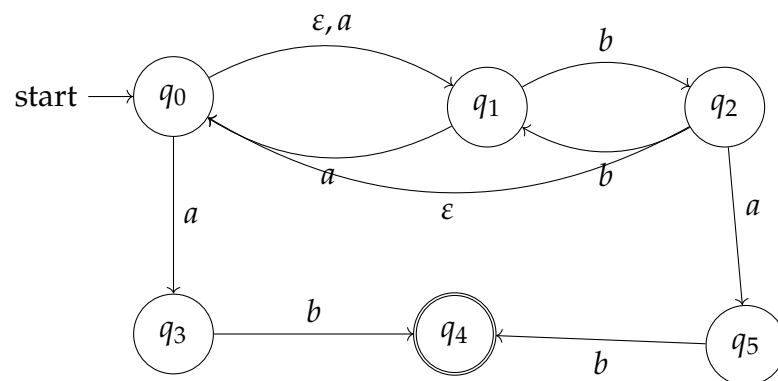
„Jede Teilsprache einer regulären Sprache ist regulär, d. h. für ein Alphabet und formale Sprachen  $L' \subseteq L$  ist  $L'$  regulär, falls  $L$  regulär ist.“

Lösungsvorschlag

Ja. Reguläre Sprachen sind abgeschlossen unter dem Komplement und der Vereinigung

## 46115 / 2021 / Frühjahr / Thema 2 / Teilaufgabe 1 / Aufgabe 2

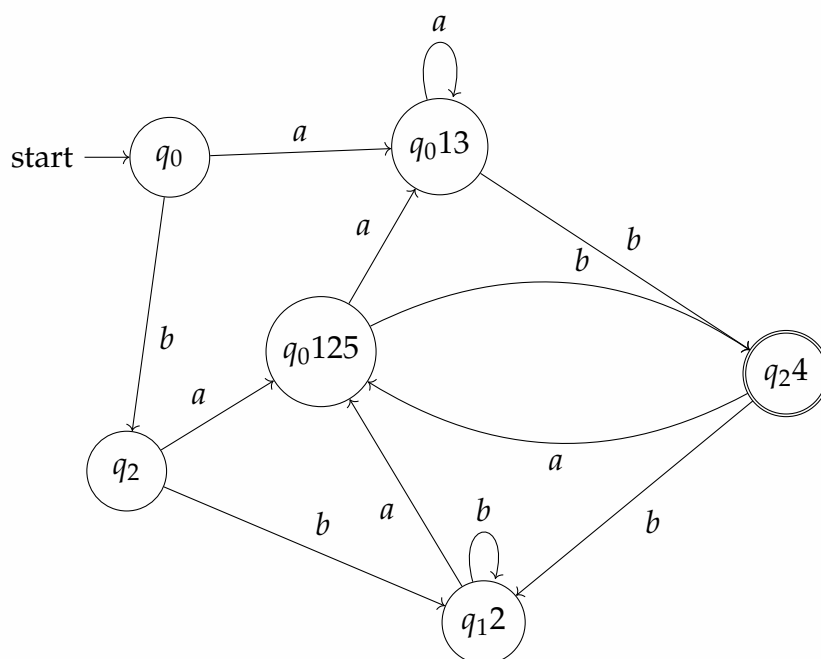
Gegeben sei der folgende e-nichtdeterministische endliche Automat A über dem Alphabet



Der Automat auf [flaci.com](http://flaci.com) (FLACI: Formale Sprachen, abstrakte Automaten, Compiler und Interpreter) Ein Projekt der Hochschule Zittau/Görlitz und der Pädagogischen Hochschule Schwyz: [flaci.com/A54bbh2iz](http://flaci.com/A54bbh2iz)

- (a) Konstruieren Sie einen deterministischen endlichen Automaten für A. Wenden Sie dafür die Potenzmengenkonstruktion an.

Lösungsvorschlag



Der Automat auf flaci.com (FLACI: Formale Sprachen, abstrakte Automaten, Compiler und Interpreter) Ein Projekt der Hochschule Zittau/Görlitz und der Pädagogischen Hochschule Schwyz: flaci.com/Arnqcysfc

Reguläre Sprache  
Kontextfreie Sprache

- (b) Beschreiben Sie die von A akzeptierte Sprache  $L(A)$  mit eigenen Worten und so einfach wie möglich.

Lösungsvorschlag

Endet auf ab, Präfix beliebig auch leer

### 46115 / 2021 / Frühjahr / Thema 2 / Teilaufgabe 1 / Aufgabe 3

Sei  $\mathbb{N}_0 = \{0, 1, 2, \dots\}$  die Menge aller natürlichen Zahlen mit 0. Betrachten Sie die folgenden Sprachen.

- (a)  $L_1 = \{a^{3n}b^{2n}a^n \mid n \in \mathbb{N}_0\}$

Lösungsvorschlag

nicht kontextfrei

- (b)  $L_2 = \{a^{3n}a^{2n}b^n \mid n \in \mathbb{N}_0\}$

Lösungsvorschlag

kontextfrei.

Der Ausdruck lässt umformen in:  $L_2 = \{a^{5n}b^n \mid n \in \mathbb{N}_0\}$

$P = \{$

$S \rightarrow aaaaaSb \mid \varepsilon$

$\}$

- (c)  $L_3 = \{(ab)^na(ba)^nb(ab)^n \mid n \in \mathbb{N}_0\}$

Lösungsvorschlag

nicht kontextfrei

Geben Sie jeweils an, ob  $L_1$ ,  $L_2$  und  $L_3$  kontextfrei und ob  $L_1$ ,  $L_2$  und  $L_3$  regulär sind. Beweisen Sie Ihre Behauptung und ordnen Sie jede Sprache in die kleinstmögliche Klasse (regulär, kontextfrei, nicht kontextfrei) ein. Für eine Einordnung in kontextfrei zeigen Sie also, dass die Sprache kontextfrei und nicht regulär ist.

Erfolgt ein Beweis durch Angabe eines Automaten, so ist eine klare Beschreibung der Funktionsweise des Automaten und der Bedeutung der Zustände erforderlich. Erfolgt der Beweis durch Angabe eines regulären Ausdrucks, so ist eine intuitive Beschreibung erforderlich. Wird der Beweis durch die Angabe einer Grammatik geführt, so ist die Bedeutung der Variablen zu erläutern.

## 66115 / 2007 / Herbst / Thema 2 / Aufgabe 1

Gegeben sei der nichtdeterministische endliche Automat  $M$  mit dem Alphabet  $\Sigma = \{a, b\}$ , der Zustandsmenge  $\{z_0, z_1, z_2, z_3\}$ , Anfangszustand  $z_0$ , Endzustand  $\{z_3\}$  und der Überföhrungsfunktion  $\delta$  mit:

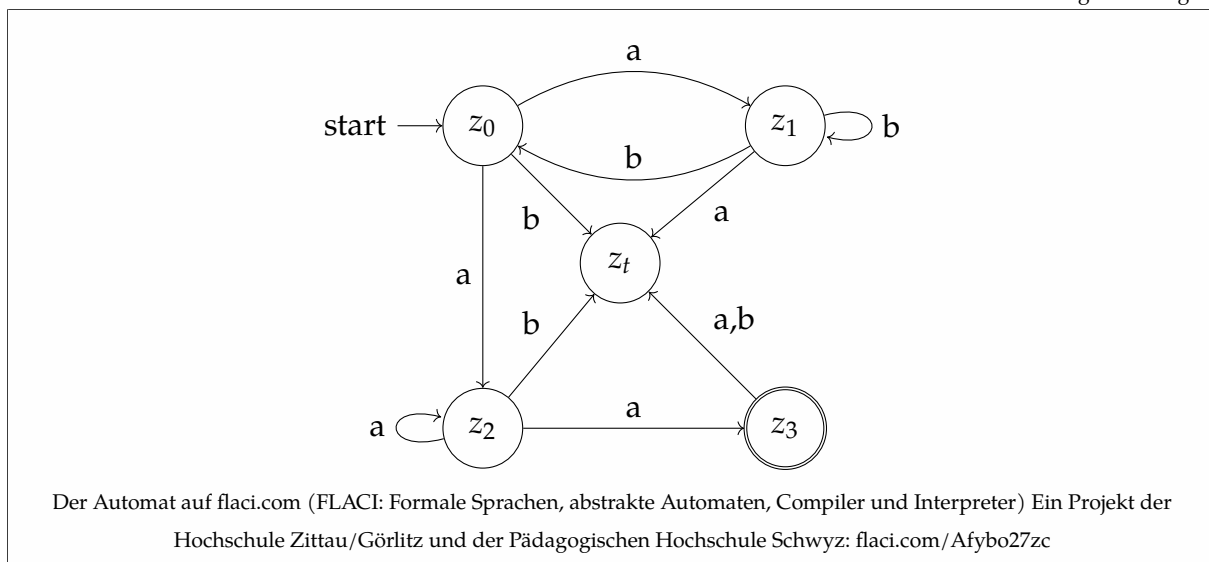
$$\delta(z_0, a) = \{z_1, z_2\},$$

$$\delta(z_1, b) = \{z_0, z_1\},$$

$$\delta(z_2, a) = \{z_2, z_3\},$$

$$\delta(z_0, b) = \delta(z_1, a) = \delta(z_2, b) = \delta(z_3, a) = \delta(z_3, b) = \emptyset$$

Lösungsvorschlag



$L(M)$  sei die von  $M$  akzeptierte Sprache.

(a) Gelten folgende Aussagen?

(i) Es gibt Zeichenreihen in  $L(M)$ , die genauso viele  $a$ 's enthalten wie  $b$ 's.

Lösungsvorschlag

Ja, zum Beispiel das Wort  $abbbbaa$  oder  $abbbbbaaa$ . Mit der Überföhrungsfunktion  $\delta(z_1, b) = \{z_1\}$  können beliebig viele  $b$ 's akzeptiert werden, sodass die Anzahl von  $a$ 's und  $b$ 's ausgeglichen werden kann.

(ii) Jede Zeichenreihe in  $L(M)$ , die mindestens vier  $b$ 's enthält, enthält auch mindestens vier  $a$ 's.

Lösungsvorschlag

Nein, z. B. das Wort  $abbbbbaa$  wird akzeptiert. Ein Wort muss nur mindestens drei  $a$ 's enthalten. Mit der Überföhrungsfunktion  $\delta(z_1, b) = \{z_1\}$  können aber beliebig viele  $b$ 's akzeptiert werden.

Begründen Sie Ihre Antworten.

(b) Geben Sie eine reguläre (Typ-3-)Grammatik an, die  $L(M)$  erzeugt.

(c) Beschreiben Sie  $L(M)$  durch einen regulären Ausdruck.

Lösungsvorschlag

$(ab+)^*aa+$

(d) Konstruieren Sie aus  $M$  mit der Potenzmengen-Konstruktion (und entsprechenden Begründung) einen deterministischen endlichen Automaten, der  $L(M)$  akzeptiert.

Lösungsvorschlag

Name	Zustandsmenge	Eingabe a	Eingabe b
$Z_0$	$Z_0 \{z_0\}$	$Z_1 \{z_1, z_2\}$	$Z_2 \{z_t\}$
$Z_1$	$Z_1 \{z_1, z_2\}$	$Z_3 \{z_2, z_3, z_t\}$	$Z_4 \{z_0, z_1, z_t\}$
$Z_2$	$Z_2 \{z_t\}$	$Z_2 \{z_t\}$	$Z_2 \{z_t\}$
$Z_3$	$Z_3 \{z_2, z_3, z_t\}$	$Z_3 \{z_2, z_3, z_t\}$	$Z_2 \{z_t\}$
$Z_4$	$Z_4 \{z_0, z_1, z_t\}$	$Z_5 \{z_1, z_2, z_t\}$	$Z_4 \{z_0, z_1, z_t\}$
$Z_5$	$Z_5 \{z_1, z_2, z_t\}$	$Z_3 \{z_2, z_3, z_t\}$	$Z_4 \{z_0, z_1, z_t\}$

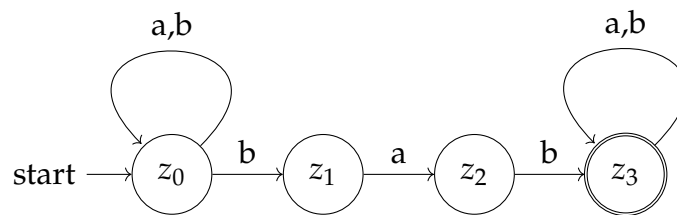
Der Automat auf flaci.com (FLACI: Formale Sprachen, abstrakte Automaten, Compiler und Interpreter) Ein Projekt der Hochschule Zittau/Görlitz und der Pädagogischen Hochschule Schwyz: flaci.com/Aig9i4u7z

## 66115 / 2012 / Herbst / Thema 1 / Aufgabe 1

Wir fixieren das Alphabet  $\Sigma = \{a, b\}$  und definieren  $L \subseteq \Sigma^*$  durch

$$L = \{w \mid \text{in } w \text{ kommt das Teilwort } bab \text{ vor}\}$$

z. B. ist  $babaabb \in L$ , aber  $baabaabb \notin L$ . Der folgende nichtdeterministische Automat  $A$  erkennt  $L$ :

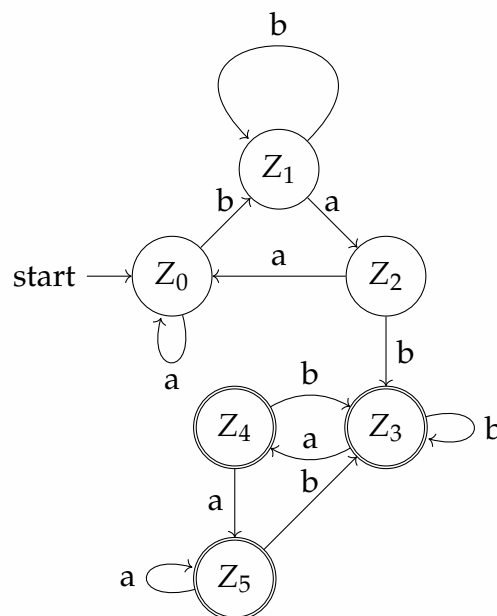


Der Automat auf flaci.com (FLACI: Formale Sprachen, abstrakte Automaten, Compiler und Interpreter) Ein Projekt der Hochschule Zittau/Görlitz und der Pädagogischen Hochschule Schwyz: flaci.com/Af75jwj3r

- (a) Wenden Sie die Potenzmengenkonstruktion auf den Automaten an und geben Sie den resultierenden deterministischen Automaten an. Nicht erreichbare Zustände sollen nicht dargestellt werden.

Lösungsvorschlag

Zustandsmenge	Eingabe a	Eingabe b
$Z_0 \{z_0\}$	$Z_0 \{z_0\}$	$Z_1 \{z_0, z_1\}$
$Z_1 \{z_0, z_1\}$	$Z_2 \{z_0, z_2\}$	$Z_1 \{z_0, z_1\}$
$Z_2 \{z_0, z_2\}$	$Z_0 \{z_0\}$	$Z_3 \{z_0, z_1, z_3\}$
$Z_3 \{z_0, z_1, z_3\}$	$Z_4 \{z_0, z_2, z_3\}$	$Z_3 \{z_0, z_1, z_3\}$
$Z_4 \{z_0, z_2, z_3\}$	$Z_5 \{z_0, z_3\}$	$Z_3 \{z_0, z_1, z_3\}$
$Z_5 \{z_0, z_3\}$	$Z_5 \{z_0, z_3\}$	$Z_3 \{z_0, z_1, z_3\}$



Der Automat auf flaci.com (FLACI: Formale Sprachen, abstrakte Automaten, Compiler und Interpreter) Ein Projekt der Hochschule Zittau/Görlitz und der Pädagogischen Hochschule Schwyz: flaci.com/Aro483e89

- (b) Konstruieren Sie aus dem so erhaltenen deterministischen Automaten den Minimalautomaten für  $L$ . Beschreiben Sie dabei die Arbeitsschritte des verwendeten



## Algorithmus in nachvollziehbarer Weise.

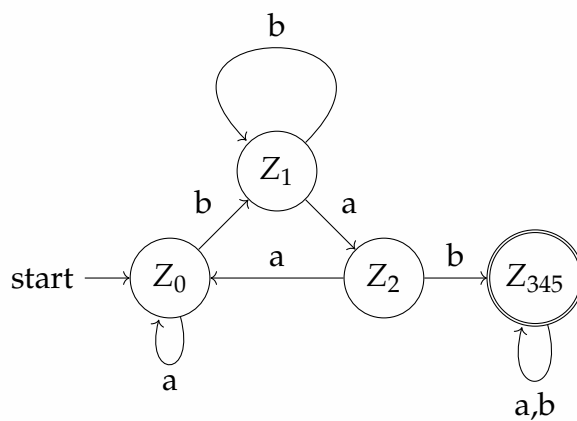
Lösungsvorschlag

$z_0$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$
$z_1$	$x_3$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$
$z_2$	$x_2$	$x_2$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$
$z_3$	$x_1$	$x_1$	$x_1$	$\emptyset$	$\emptyset$	$\emptyset$
$z_4$	$x_1$	$x_1$	$x_1$		$\emptyset$	$\emptyset$
$z_5$	$x_1$	$x_1$	$x_1$			$\emptyset$
	$z_0$	$z_1$	$z_2$	$z_3$	$z_4$	$z_5$

- $x_1$  Paar aus End- / Nicht-Endzustand kann nicht äquivalent sein.  
 $x_2$  Test, ob man mit der Eingabe zu einem bereits markiertem Paar kommt.  
 $x_3$  In weiteren Iterationen markierte Zustände.  
 $x_4$  ...

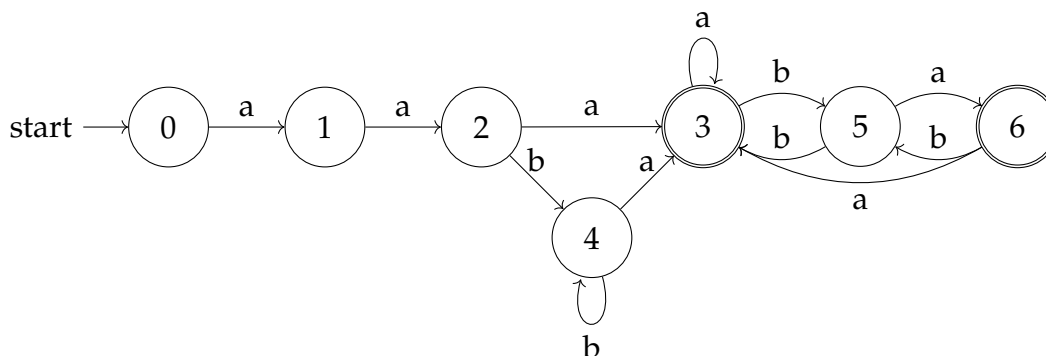
## Übergangstabelle

Zustandspaar	a	b
$(z_0, z_1)$	$(z_0, z_2) \ x_3$	$(z_1, z_1)$
$(z_0, z_2)$	$(z_0, z_0)$	$(z_1, z_3) \ x_2$
$(z_1, z_2)$	$(z_2, z_0) \ x_3$	$(z_1, z_3) \ x_2$
$(z_3, z_4)$	$(z_4, z_5)$	$(z_3, z_3)$
$(z_3, z_5)$	$(z_4, z_5)$	$(z_3, z_3)$
$(z_4, z_5)$	$(z_5, z_5)$	$(z_3, z_3)$



## 66115 / 2013 / Herbst / Thema 2 / Aufgabe 3

Minimieren Sie den folgenden deterministischen Automaten mit den Zuständen  $\{0, 1, 2, 3, 4, 5, 6\}$ , dem Startzustand 0 und den Endzuständen  $\{3, 6\}$ . Geben Sie z. B. durch die Bezeichnung an, welche Zustände zusammengefasst wurden.



Lösungsvorschlag

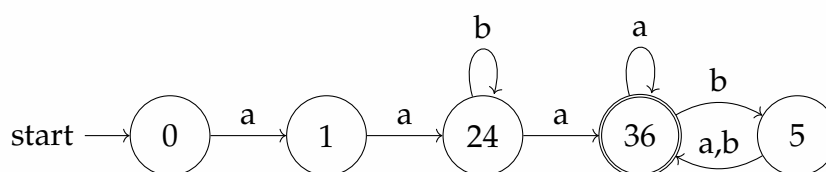
0	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$
1	$x_3$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$
2	$x_2$	$x_2$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$
3	$x_1$	$x_1$	$x_1$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$
4	$x_2$	$x_2$		$x_1$	$\emptyset$	$\emptyset$	$\emptyset$
5	$x_2$	$x_2$	$x_2$	$x_1$	$x_2$	$\emptyset$	$\emptyset$
6	$x_1$	$x_1$	$x_1$		$x_1$	$x_1$	$\emptyset$
	0	1	2	3	4	5	6

- $x_1$  Paar aus End-/ Nicht-Endzustand kann nicht äquivalent sein.  
 $x_2$  Test, ob man mit der Eingabe zu einem bereits markiertem Paar kommt.  
 $x_3$  In weiteren Iterationen markierte Zustände.  
 $x_4$  ...

### Übergangstabelle

Zustandspaar	a	b
(0, 1)	(1, 2) $x_3$	(T, T)
(0, 2)	(1, 3) $x_2$	(T, 4)
(0, 4)	(1, 3) $x_2$	(T, 4)
(0, 5)	(1, 6) $x_2$	(T, 3)
(1, 2)	(2, 3) $x_2$	(T, 4)
(1, 4)	(2, 3) $x_2$	(T, 4)
(1, 5)	(2, 6) $x_2$	(T, 3)
(2, 4)	(3, 3)	(4, 4)
(2, 5)	(3, 6)	(3, 4) $x_2$
(3, 6)	(3, 3)	(5, 5)
(4, 5)	(3, 6)	(3, 4) $x_2$

T = Trap-Zustand = Falle



## 66115 / 2015 / Frühjahr / Thema 1 / Aufgabe 1

Die Sprache  $L$  über den Alphabet  $\Sigma = \{0, 1\}$  enthält alle Wörter, bei denen beim Lesen von links nach rechts der Unterschied in der Zahl der 0en und 1en stets höchstens 3 ist. Also ist  $w \in L$  genau dann, wenn für alle  $u, v$  mit  $w = uv$  gilt  $||u|_0 - |u|_1| \leq 3$ . Erinnerung:  $|w|_a$  bezeichnet die Zahl der  $a$ 's im Wort  $w$ .

- (a) Sei  $A = (Q, \Sigma, \delta, q_0, E)$  ein deterministischer endlicher Automat für  $L$ . Es sei  $w_1 = 111$ ,  $w_2 = 11$ ,  $w_3 = 1$ ,  $w_4 = \varepsilon$ ,  $w_5 = 0$ ,  $w_6 = 00$ ,  $w_7 = 000$ . Machen Sie sich klar, dass der Automat jedes dieser Wörter verarbeiten können muss. Folgern Sie, dass der Automat mindestens sieben Zustände haben muss. Schreiben Sie Ihr Argumentation schlüssig und vollständig auf.

Ein deterministischer endlicher Automat hat keinen zusätzlichen Speicher zur Verfügung, in dem die Anzahl der bisher vorkommenden Einsen und Nullen gespeichert werden könnte. Ein deterministischer endlicher Automat kann die von der Sprache benötigten Anzahl an Einsen und Nullen nur in Form von Zuständen speichern. Um die Anzahl von 3 Einsen bzw. 3 Nullen zu speichern, sind also 6 Zustände nötig.

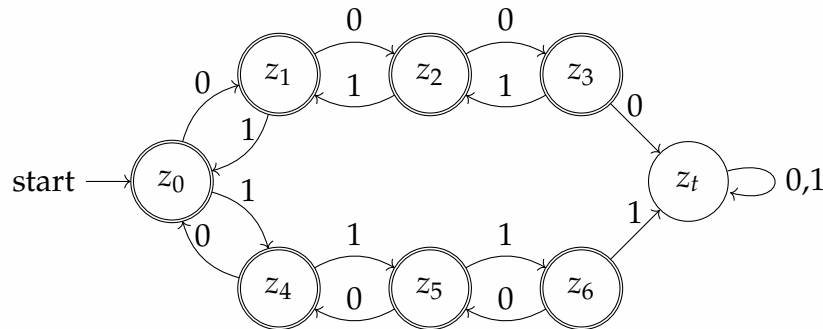
Die Wörter 01 oder 0011 oder 0101 etc. haben eine Differenz von 0, wenn die

Anzahl an Nullen und Einsen abgezogen wird. Um auch diese Wörter darstellen zu können, ist mindestens ein weiterer Zustand nötig.

## Pumping-Lemma (Reguläre Sprache)

(b) Begründen Sie, dass  $L$  regulär ist.

## Lösungsvorschlag



Der Automat auf flaci.com (FLACI: Formale Sprachen, abstrakte Automaten, Compiler und Interpreter) Ein Projekt der Hochschule Zittau/Görlitz und der Pädagogischen Hochschule Schwyz: [flaci.com/Ait6ya31c](http://flaci.com/Ait6ya31c)

(c) Jemand behauptet, diese Sprache sei nicht regulär und gibt folgenden „Beweis“ dafür an: Wäre  $L$  regulär, so sei  $n$  eine entsprechende Pumping-Zahl. Nun ist  $w = (01)^n \in L$ . Zerlegt man nun  $w = uxv$ , wobei  $u = 0, x = 1, v = (01)^{n-1}$ , so ist zum Beispiel  $ux^5v \notin L$ , denn es ist  $ux^5v = 01111101010101\dots$ . Legen Sie genau dar, an welcher Stelle dieser „Beweis“ fehlerhaft ist.

## Exkurs: Pumping-Lemma für Reguläre Sprachen

Es sei  $L$  eine reguläre Sprache. Dann gibt es eine Zahl  $j$ , sodass für alle Wörter  $\omega \in L$  mit  $|\omega| \geq j$  (jedes Wort  $\omega$  in  $L$  mit Mindestlänge  $j$ ) jeweils eine Zerlegung  $\omega = uvw$  existiert, sodass die folgenden Eigenschaften erfüllt sind:

- (i)  $|v| \geq 1$  (Das Wort  $v$  ist nicht leer.)
- (ii)  $|uv| \leq j$  (Die beiden Wörter  $u$  und  $v$  haben zusammen höchstens die Länge  $j$ .)
- (iii) Für alle  $i = 0, 1, 2, \dots$  gilt  $uv^i w \in L$  (Für jede natürliche Zahl (mit 0)  $i$  ist das Wort  $uv^i w$  in der Sprache  $L$ )

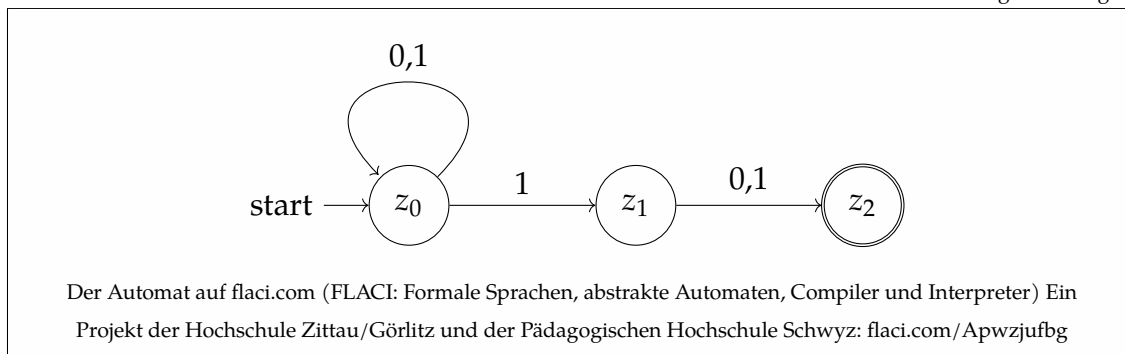
Die kleinste Zahl  $j$ , die diese Eigenschaften erfüllt, wird Pumping-Zahl der Sprache  $L$  genannt.

Lösungsvorschlag

Das Wort  $(01)^n$  wurde falsch zerlegt. Für die Pumping-Zahl  $n = 3$  gibt es sehr wohl eine Zerlegung, die beim Aufpumpen regulär ist, also:  $\omega = 010101$  ( $u = 01$ ,  $x = 01$  und  $v = 01$ ).  $ux^5v = 01010101010101 \in L$ . Es gibt also eine Zerlegung, die beim Aufpumpen die 3 Pumping-Lemma-Eigenschaften erfüllt. Daher kann man das Pumping-Lemma so nicht widerlegt werden, indem man ein einziges Gegenbeispiel gibt.

- (d) In anderen Fällen können nichtdeterministische endliche Automaten echt kleiner sein als die besten deterministischen Automaten. Ein Beispiel ist die Sprache  $L_2 = \Sigma^*1\Sigma$  aller Wörter, deren vorletztes Symbol 1 ist. Geben Sie einen nicht-deterministischen Automaten mit nur drei Zuständen an,  $L_2$  erkennt.

Lösungsvorschlag

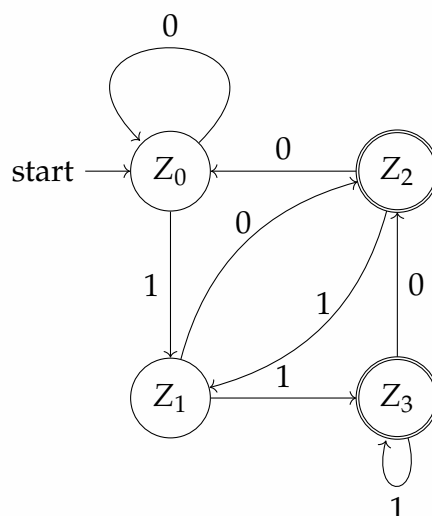


- (e) Führen Sie auf Ihrem Automaten die Potenzmengenkonstruktion und anschließend den Minimierungsalgorithmus durch. Wie viele Zustände muss ein deterministischer Automat für  $L_2$  also mindestens haben?

Lösungsvorschlag

**Potenzmengenkonstruktion**

Name	Zustandsmenge	Eingabe 0	Eingabe 1
$Z_0$	$Z_0 \{z_0\}$	$Z_0 \{z_0\}$	$Z_1 \{z_0, z_1\}$
$Z_1$	$Z_1 \{z_0, z_1\}$	$Z_2 \{z_0, z_2\}$	$Z_3 \{z_0, z_1, z_2\}$
$Z_2$	$Z_2 \{z_0, z_2\}$	$Z_0 \{z_0\}$	$Z_1 \{z_0, z_1\}$
$Z_3$	$Z_3 \{z_0, z_1, z_2\}$	$Z_2 \{z_0, z_2\}$	$Z_3 \{z_0, z_1, z_2\}$



Der Automat auf flaci.com (FLACI: Formale Sprachen, abstrakte Automaten, Compiler und Interpreter) Ein Projekt der Hochschule Zittau/Görlitz und der Pädagogischen Hochschule Schwyz: flaci.com/Ajfcfb9

## Minimierungsalgorithmus

$Z_0$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$
$Z_1$	$x_2$	$\emptyset$	$\emptyset$	$\emptyset$
$Z_2$	$x_1$	$x_1$	$\emptyset$	$\emptyset$
$Z_3$	$x_1$	$x_1$	$x_2$	$\emptyset$
	$Z_0$	$Z_1$	$Z_2$	$Z_3$

- $x_1$  Paar aus End- / Nicht-Endzustand kann nicht äquivalent sein.  
 $x_2$  Test, ob man mit der Eingabe zu einem bereits markiertem Paar kommt.  
 $x_3$  In weiteren Iterationen markierte Zustände.  
 $x_4$  ...

## Übergangstabelle

Zustandspaar	0	1
$(Z_0, Z_1)$	$(Z_0, Z_2) \ x_2$	$(Z_1, Z_3) \ x_2$
$(Z_2, Z_3)$	$(Z_0, Z_1)$	$(Z_1, Z_3) \ x_2$

Wie aus der oben stehenden Tabelle abzulesen ist, gibt es keine äquivalenten Zustände. Der Automat kann nicht minimiert werden. Er ist bereits minimal.

## 66115 / 2016 / Frühjahr / Thema 1 / Aufgabe 1

- (a) Geben Sie einen möglichst einfachen regulären Ausdruck für die Sprache  $L_1 = \{a_1, a_2, \dots, a_n \mid n \geq 3, a_i \in \{a, b\} \text{ für alle } i = 1, \dots, n \text{ und } a_1 \geq a_n\}$  an.

Lösungsvorschlag

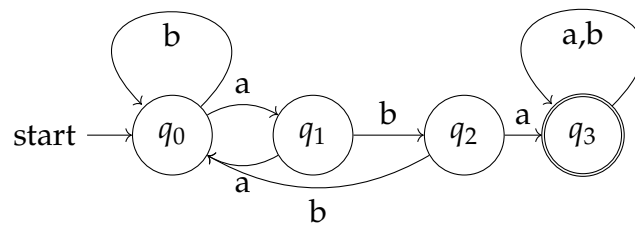
$((a(a|b)+b) | (b(a|b)+a))$

- (b) Geben Sie einen möglichst einfachen regulären Ausdruck für die Sprache  $L_2 = \{w \in \{a, b\}^* \mid w \text{ enthält genau ein } b \text{ und ist von ungerader Länge}\}$  an.

Lösungsvorschlag

$(aa)^*(b|aba)(aa)^*$

- (c) Beschreiben Sie die Sprache des folgenden Automaten  $A_1$ , möglichst einfach und präzise in ihren eigenen Worten.

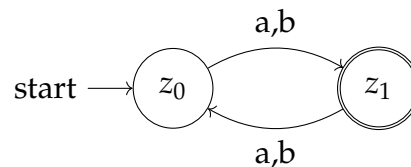


Der Automat auf flaci.com (FLACI: Formale Sprachen, abstrakte Automaten, Compiler und Interpreter) Ein Projekt der Hochschule Zittau/Görlitz und der Pädagogischen Hochschule Schwyz: [flaci.com/Arz003ccg](http://flaci.com/Arz003ccg)

Lösungsvorschlag

Die Sprache enthält das Teilwort *aba*

(d) Betrachten Sie folgenden Automaten  $A_2$ :



Der Automat auf flaci.com (FLACI: Formale Sprachen, abstrakte Automaten, Compiler und Interpreter) Ein Projekt der Hochschule Zittau/Görlitz und der Pädagogischen Hochschule Schwyz: [flaci.com/Ap9qbkumc](http://flaci.com/Ap9qbkumc)

Im Original sind die Zustände mit  $q_x$  benannt. Damit wir die Schnittmenge besser bilden können, wird hier  $z_x$  verwendet.

Konstruieren Sie einen endlichen Automaten, der die Schnittmenge der Sprachen  $L(A_1)$  und  $L(A_2)$  akzeptiert.

Lösungsvorschlag

$A_1$

	a	b
$q_0$	$q_1$	$q_0$
$q_1$	$q_0$	$q_2$
$q_2$	$q_3$	$q_0$
$q_3$	$q_3$	$q_3$

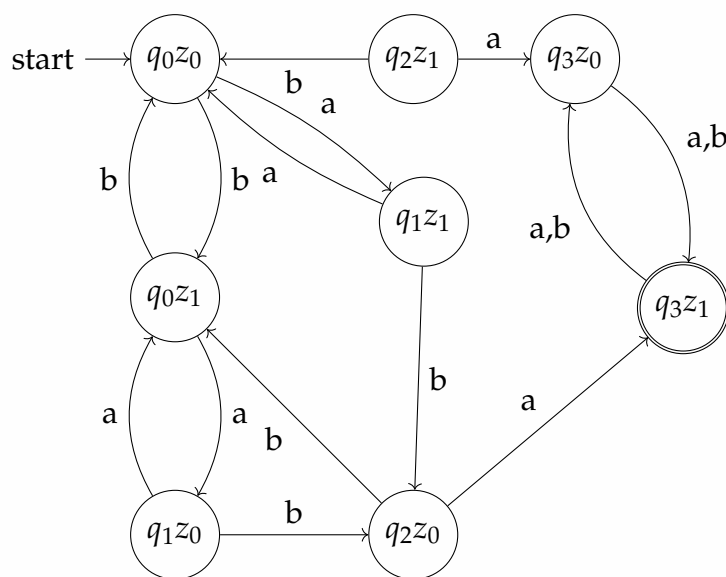
$A_2$

	a	b
$z_0$	$z_1$	$z_1$
$z_1$	$z_0$	$z_0$

Neuer Endzustand:  $q_3z_1$



	a	b
$q_0z_0$	$q_1z_1$	$q_0z_1$
$q_1z_0$	$q_0z_1$	$q_2z_1$
$q_2z_0$	$q_3z_1$	$q_0z_1$
$q_3z_0$	$q_3z_1$	$q_3z_1$
$q_0z_1$	$q_1z_0$	$q_0z_0$
$q_1z_1$	$q_0z_0$	$q_2z_0$
$q_2z_1$	$q_3z_0$	$q_0z_0$
$q_3z_1$	$q_3z_0$	$q_3z_0$



Der Automat auf flaci.com (FLACI: Formale Sprachen, abstrakte Automaten, Compiler und Interpreter) Ein Projekt der Hochschule Zittau/Görlitz und der Pädagogischen Hochschule Schwyz: [flaci.com/Ar3pc5rh7](http://flaci.com/Ar3pc5rh7)

## 66115 / 2016 / Herbst / Thema 1 / Aufgabe 1

Gegeben ist der deterministische endliche Automat  $A = (\{A, B, C, D, E\}, \{0, 1\}, \delta, \{E\}, A)$ , wobei

- Minimieren Sie den Automaten mit dem bekannten Minimierungsalgorithmus. Dokumentieren Sie die Schritte geeignet.
- Geben Sie einen regulären Ausdruck für die erkannte Sprache an.
- Geben Sie die Äquivalenzklassen der Myhill-Nerode-Äquivalenz der Sprache durch reguläre Ausdrücke an.

- (d) Geben Sie ein Beispiel einer regulären Sprache an, für die kein deterministischer endlicher Automat mit höchstens zwei Endzuständen existiert.
- (e) Geben Sie ein Beispiel einer regulären Sprache an, für die kein deterministischer endlicher Automat mit weniger als fünf Zuständen existiert.

### 66115 / 2018 / Frühjahr / Thema 2 / Aufgabe 3

Gesucht ist eine reguläre Sprache  $C \subseteq \{a, b\}^*$ , deren minimaler deterministischer endlicher Automat (DEA) mindestens 4 Zustände mehr besitzt als der minimale nichtdeterministische endliche Automat (NEA). Gehen Sie wie folgt vor:

- (a) Definieren Sie  $C \subseteq \{a, b\}^*$  und erklären Sie kurz, warum es bei dieser Sprache NEAs gibt, die deutlich kleiner als der minimale DEA sind.

Lösungsvorschlag

Sprache mit exponentiellem Blow-Up:

Ein NEA der Sprache

$$\begin{aligned} L_k &= \{xay \mid x, y \in \{a, b\}^* \wedge |y| = k - 1\} \\ &= \{w \in \{a, b\}^* \mid \text{der } k\text{-te Buchstabe von hinten ist ein } a\} \end{aligned}$$

kommt mit  $k + 1$  Zuständen aus.

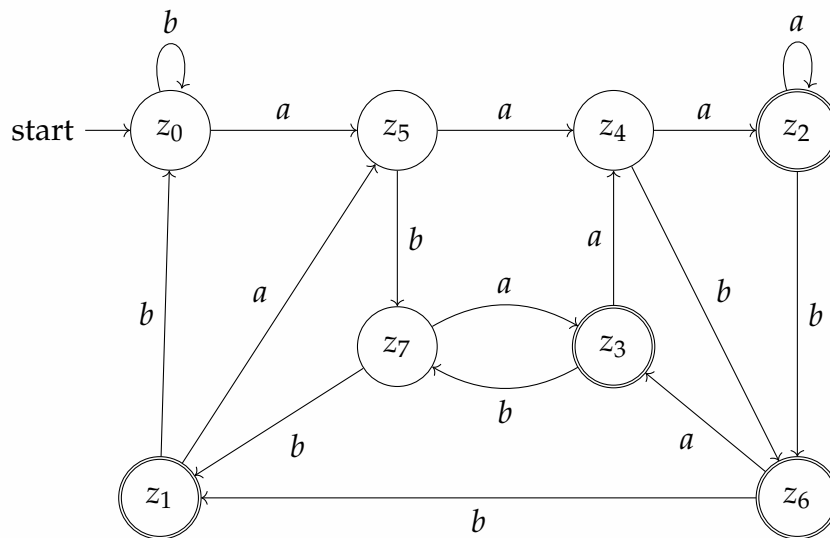
Jeder DEA  $M$  mit  $L(M) = L$  hat dann mindestens  $2^k$  Zustände. Wir wählen  $k = 3$ . Dann hat der zugehörige NEA 4 Zustände und der zugehörige DEA mindestens 8. Sei also  $L_k = \{xay \mid x, y \in \{a, b\}^* \wedge |y| = 2\}$  die gesuchte Sprache.

Der informelle Grund, warum ein DEA für die Sprache  $L_k$  groß sein muss, ist dass er sich immer die letzten  $n$  Symbole merken muss.<sup>a</sup>

<sup>a</sup><https://www.tcs.ifi.lmu.de/lehre/ss-2013/timi/handouts/handout-02>

- (b) Geben Sie den minimalen DEA  $M$  für  $C$  an. (Zeichnung des DEA genügt; die Minimalität muss nicht begründet werden.)

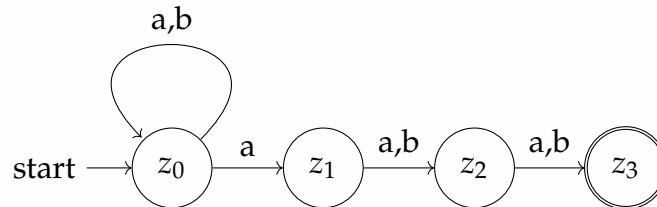
Lösungsvorschlag



Der Automat auf flaci.com (FLACI: Formale Sprachen, abstrakte Automaten, Compiler und Interpreter) Ein Projekt der Hochschule Zittau/Görlitz und der Pädagogischen Hochschule Schwyz: [flaci.com/Ahhefpjir](http://flaci.com/Ahhefpjir)

- (c) Geben Sie einen NEA  $N$  für  $C$  an, der mindestens 4 Zustände weniger besitzt als  $M$ . (Zeichnung des NEA genügt)

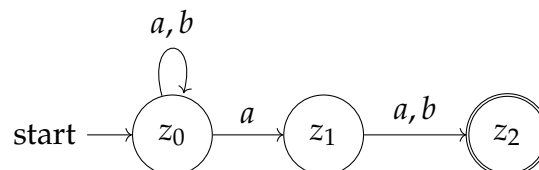
Lösungsvorschlag



Der Automat auf flaci.com (FLACI: Formale Sprachen, abstrakte Automaten, Compiler und Interpreter) Ein Projekt der Hochschule Zittau/Görlitz und der Pädagogischen Hochschule Schwyz: [flaci.com/Ajrz7h5r7](http://flaci.com/Ajrz7h5r7)

## 66115 / 2019 / Frühjahr / Thema 1 / Aufgabe 2

- (a) Gegeben sei der nichtdeterministische endliche Automat  $A$  über dem Alphabet  $\Sigma = \{a, b\}$  wie folgt:



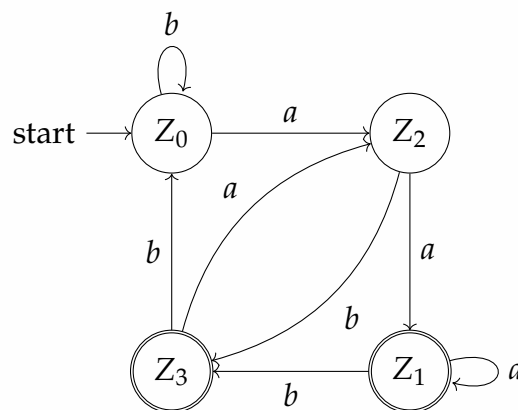
Der Automat auf flaci.com (FLACI: Formale Sprachen, abstrakte Automaten, Compiler und Interpreter) Ein Projekt der Hochschule Zittau/Görlitz und der Pädagogischen Hochschule Schwyz: [flaci.com/Arozq4rm2](http://flaci.com/Arozq4rm2)

Konstruieren Sie einen deterministischen endlichen Automaten, der das Komplement  $\bar{L}(A) = \{ w \in \Sigma^* \mid w \notin L(A) \}$  der von  $A$  akzeptierten Sprache  $L(A)$  akzeptiert.

Lösungsvorschlag

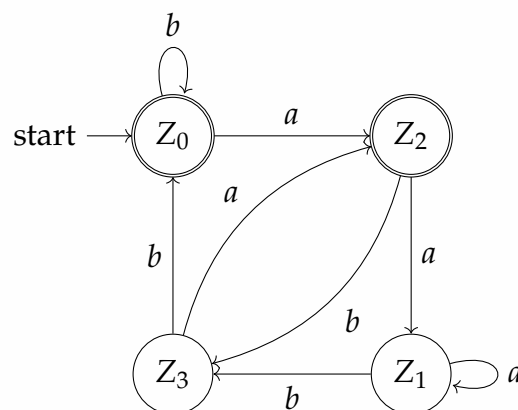
Wir konvertieren zuerst den nichtdeterministischen endlichen Automaten in einen deterministischen endlichen Automaten mit Hilfe des Potenzmengenalgorithmus.

Zustandsmenge	Eingabe $a$	Eingabe $b$
$Z_0 \{z_0\}$	$Z_1 \{z_0, z_1\}$	$Z_0 \{z_0\}$
$Z_1 \{z_0, z_1\}$	$Z_2 \{z_0, z_1, z_2\}$	$Z_3 \{z_0, z_2\}$
$Z_2 \{z_0, z_1, z_2\}$	$Z_2 \{z_0, z_1, z_2\}$	$Z_3 \{z_0, z_2\}$
$Z_3 \{z_0, z_2\}$	$Z_1 \{z_0, z_1\}$	$Z_0 \{z_0\}$



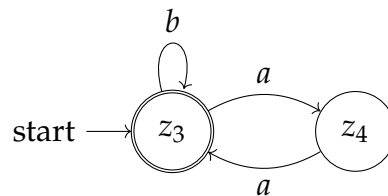
Der Automat auf flaci.com (FLACI: Formale Sprachen, abstrakte Automaten, Compiler und Interpreter) Ein Projekt der Hochschule Zittau/Görlitz und der Pädagogischen Hochschule Schwyz: [flaci.com/Arxujcbdg](http://flaci.com/Arxujcbdg)

Wir vertauschen die End- und Nicht-End-Zustände, um das Komplement zu erhalten:



Der Automat auf flaci.com (FLACI: Formale Sprachen, abstrakte Automaten, Compiler und Interpreter) Ein Projekt der Hochschule Zittau/Görlitz und der Pädagogischen Hochschule Schwyz: flaci.com/A5zqsonq2

- (b) Gegeben sei zudem der nichtdeterministische Automat  $B$  über dem Alphabet  $\Sigma = \{a, b\}$ :

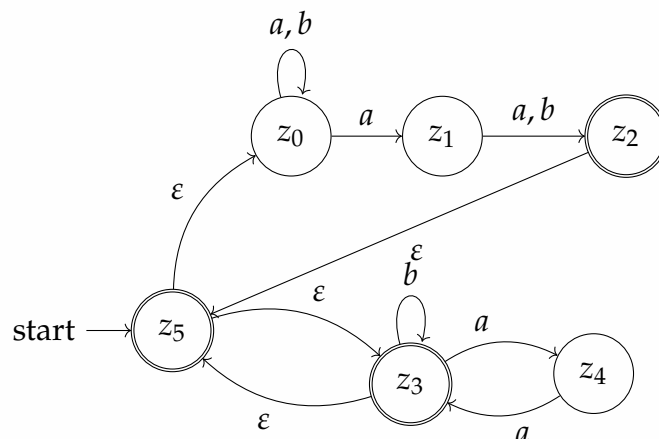


Der Automat auf flaci.com (FLACI: Formale Sprachen, abstrakte Automaten, Compiler und Interpreter) Ein Projekt der Hochschule Zittau/Görlitz und der Pädagogischen Hochschule Schwyz: flaci.com/Arafk0h2

Konstruieren Sie einen endlichen Automaten (möglicherweise mit  $\varepsilon$ -Übergängen), der die Sprache  $(L(A)L(B))^* \subseteq \Sigma^*$  akzeptiert ( $A$  aus der vorigen Aufgabe). Erläutern Sie auch Ihre Konstruktionsidee.

Lösungsvorschlag

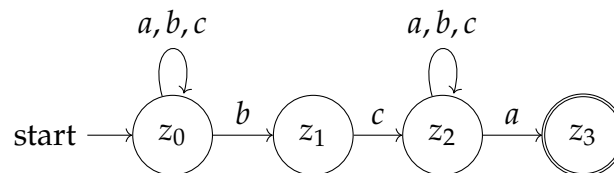
$L(A)L(B)^*$  ist die beliebige Konkatenation (Verknüpfung/Verkettung) der Sprachen  $L(A)$  und  $L(B)$  mit dem leeren Wort. Wir führen einen neuen Startzustand ( $z_5$ ) ein, der zugleich Endzustand ist. Dadurch wird das leere Wort akzeptiert. Dieser neue Startzustand führt über  $\varepsilon$ -Übergängen zu den ehemaligen Startzuständen der Automaten  $A$  und  $B$ . Die Endzustände der Automaten  $A$  und  $B$  führen über  $\varepsilon$ -Übergängen zu  $z_5$ . Dadurch sind beliebige Konkatenationen möglich.



Der Automat auf flaci.com (FLACI: Formale Sprachen, abstrakte Automaten, Compiler und Interpreter) Ein Projekt der Hochschule Zittau/Görlitz und der Pädagogischen Hochschule Schwyz: flaci.com/Aro3uhzjz

## 66115 / 2020 / Frühjahr / Thema 1 / Aufgabe 2

- (a) Es sei  $L \subseteq \{a, b, c\}^*$  die von dem folgenden nichtdeterministischen Automaten akzeptierte Sprache:



Der Automat auf flaci.com (FLACI: Formale Sprachen, abstrakte Automaten, Compiler und Interpreter) Ein Projekt der Hochschule Zittau/Görlitz und der Pädagogischen Hochschule Schwyz: [flaci.com/Apmac9bwc](http://flaci.com/Apmac9bwc)

Beschreiben Sie (in Worten) wie die Wörter aus der Sprache  $L$  aussehen.

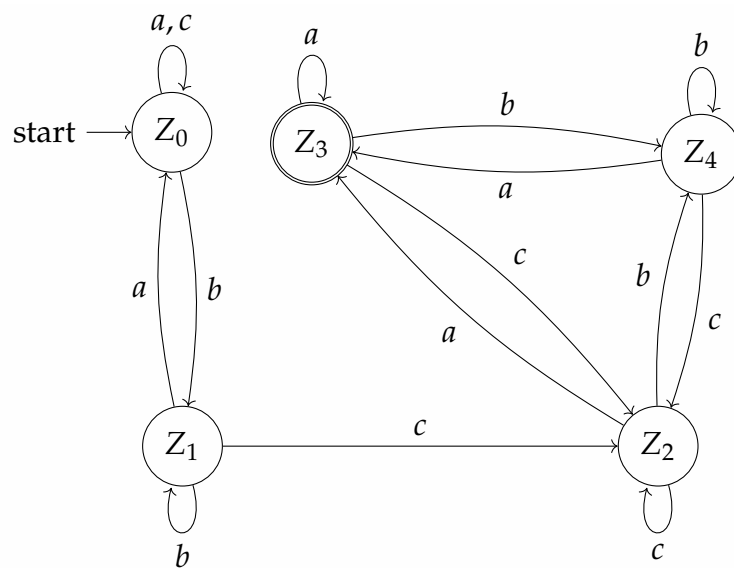
Lösungsvorschlag

Alle Wörter der Sprache  $L$  enthalten die Symbolfolge  $bc$  und enden auf  $a$ . Am Anfang der Wörter und vor dem letzten  $a$  können beliebige Kombination aus  $a, b, c$  vorkommen.

- (b) Benutzen Sie die Potenzmengenkonstruktion, um einen deterministischen Automaten zu konstruieren, der zu dem Automaten aus Teil (a) äquivalent ist. (Berechnen Sie nur erreichbare Zustände.)

Lösungsvorschlag

Zustandsmenge	Eingabe $a$	Eingabe $b$	Eingabe $c$
$Z_0 \{z_0\}$	$Z_0 \{z_0\}$	$Z_1 \{z_0, z_1\}$	$Z_0 \{z_0\}$
$Z_1 \{z_0, z_1\}$	$Z_0 \{z_0\}$	$Z_1 \{z_0, z_1\}$	$Z_2 \{z_0, z_2\}$
$Z_2 \{z_0, z_2\}$	$Z_3 \{z_0, z_2, z_3\}$	$Z_4 \{z_0, z_1, z_2\}$	$Z_2 \{z_0, z_2\}$
$Z_3 \{z_0, z_2, z_3\}$	$Z_3 \{z_0, z_2, z_3\}$	$Z_4 \{z_0, z_1, z_2\}$	$Z_2 \{z_0, z_2\}$
$Z_4 \{z_0, z_1, z_2\}$	$Z_3 \{z_0, z_2, z_3\}$	$Z_4 \{z_0, z_1, z_2\}$	$Z_2 \{z_0, z_2\}$



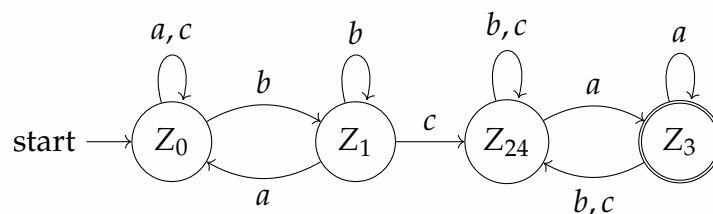
Der Automat auf flaci.com (FLACI: Formale Sprachen, abstrakte Automaten, Compiler und Interpreter) Ein Projekt der Hochschule Zittau/Görlitz und der Pädagogischen Hochschule Schwyz: [flaci.com/A5o6pho8c](http://flaci.com/A5o6pho8c)

- (c) Ist der resultierende deterministische Automat schon minimal? Begründen Sie Ihre Antwort.

Lösungsvorschlag

Nein.  $Z_2 \{z_0, z_2\}$  und  $Z_4 \{z_0, z_1, z_2\}$  können vereinigt werden, da sie bei denselben Eingaben auf die selben Potenzenmengen übergehen.

Zustandsmenge	Eingabe $a$	Eingabe $b$	Eingabe $c$
$Z_2 \{z_0, z_2\}$	$Z_3 \{z_0, z_2, z_3\}$	$Z_4 \{z_0, z_1, z_2\}$	$Z_2 \{z_0, z_2\}$
$Z_3 \{z_0, z_2, z_3\}$	$Z_3 \{z_0, z_2, z_3\}$	$Z_4 \{z_0, z_1, z_2\}$	$Z_2 \{z_0, z_2\}$
$Z_4 \{z_0, z_1, z_2\}$	$Z_3 \{z_0, z_2, z_3\}$	$Z_4 \{z_0, z_1, z_2\}$	$Z_2 \{z_0, z_2\}$



Der Automat auf flaci.com (FLACI: Formale Sprachen, abstrakte Automaten, Compiler und Interpreter) Ein Projekt der Hochschule Zittau/Görlitz und der Pädagogischen Hochschule Schwyz: [flaci.com/Ai1hox2b7](http://flaci.com/Ai1hox2b7)

- (d) Minimieren Sie den folgenden deterministischen Automaten:

**66115 / 2020 / Herbst / Thema 1 / Teilaufgabe 1 / Aufgabe 1**

Antworten Sie mit „*Stimmt*“ oder „*Stimmt nicht*“. Begründen Sie Ihr Urteil kurz.

- (a) Eine Sprache ist genau dann regulär, wenn sie unendlich viele Wörter enthält.

Lösungsvorschlag

Stimmt nicht. Sprachen mit endlicher Mächtigkeit sind immer regulär. Endliche Sprachen sind in obenstehender Aussage ausgeschlossen.

- (b) Zu jedem nichtdeterministischen endlichen Automaten mit  $n$  Zuständen gibt es einen deterministischen endlichen Automaten, der die gleiche Sprache erkennt und höchstens  $n^2$  Zustände hat.

Lösungsvorschlag

Stimmt nicht. Müsste  $2^n$  heißen.

- (c) Das Komplement einer kontextfreien Sprache ist wieder kontextfrei.

Lösungsvorschlag

Stimmt nicht. Kontextfreie Sprachen sind nicht abgeschlossen unter dem Komplement. Das Komplement einer kontextfreien Sprache kann regulär, kontextfrei oder kontextsensitiv sein.

- (d) Wenn ein Problem unentscheidbar ist, dann ist es nicht semientscheidbar.

Lösungsvorschlag

Stimmt nicht. Semientscheidbarkeit ist eine typische Form der Unentscheidbarkeit. Unentscheidbarkeit ist das Gegenteil von Entscheidbarkeit. Unentscheidbar kann entweder völlig unentscheidbar sein oder semientscheidbar.

- (e) Sei  $f$  eine totale Funktion. Dann gibt es ein WHILE-Programm, das diese berechnet.

Lösungsvorschlag

Stimmt nicht. Wir wissen nicht, ob die totale Funktion  $f$  berechenbar ist. Wenn  $f$  berechenbar ist, dann wäre die Aussage richtig.

- (f) Das Halteproblem für LOOP-Programme ist entscheidbar.

Lösungsvorschlag

Stimmt. Alle LOOP-Programme terminieren (halten). Es gibt für jede Eingabe eine Ausgabe.

- (g) Die Komplexitätsklasse  $\mathcal{NP}$  enthält genau die Entscheidungsprobleme, die in nicht-polynomieller Zeit entscheidbar sind.



Stimmt. Die Aussage entspricht der Definition der Komplexitätsklasse  $\mathcal{NP}$ .

- (h) Falls  $P \geq NP$ , dann gibt es keine  $\mathcal{NP}$ -vollständigen Probleme, die in  $P$  liegen.

Stimmt. Entspricht der Definition.

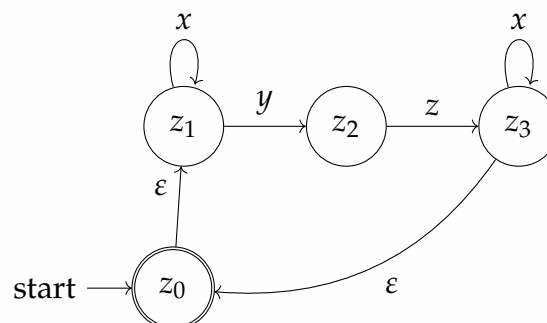
## 66115 / 2020 / Herbst / Thema 1 / Teilaufgabe 1 / Aufgabe 2

Sei  $\Sigma = \{x, y, z\}$ . Sei  $L = (x^* y z x^*)^* \subseteq \Sigma^*$ .

- (a) Geben Sie einen endlichen (deterministischen oder nichtdeterministischen) Automaten  $A$  an, der  $L$  erkennt bzw. akzeptiert.

### Nichtdeterministischer endlicher Automat

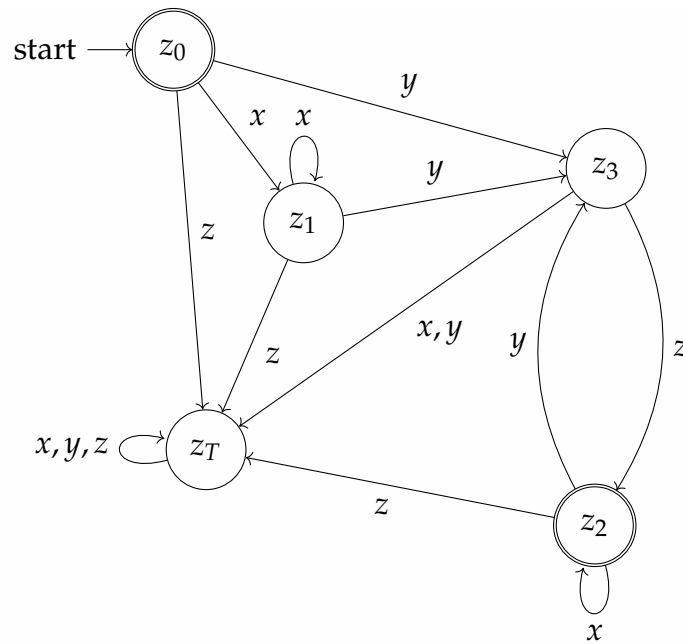
$$A_{\text{NEA}} = (\{z_0, z_1, z_2, z_3, z_T\}, \{x, y, z\}, \delta, \{z_0, z_2\}, z_0)$$



Der Automat auf flaci.com (FLACI: Formale Sprachen, abstrakte Automaten, Compiler und Interpreter) Ein Projekt der Hochschule Zittau/Görlitz und der Pädagogischen Hochschule Schwyz: [flaci.com/Ajpmxqvh9](http://flaci.com/Ajpmxqvh9)

### Deterministischer endlicher Automat

$$A_{\text{DEA}} = (\{z_0, z_1, z_2, z_3, z_T\}, \{x, y, z\}, \delta, \{z_0, z_2\}, z_0)$$



Der Automat auf flaci.com (FLACI: Formale Sprachen, abstrakte Automaten, Compiler und Interpreter) Ein Projekt der Hochschule Zittau/Görlitz und der Pädagogischen Hochschule Schwyz: flaci.com/A5xo470g9

(b) Geben Sie eine reguläre und eindeutige Grammatik  $G$  an, die  $L$  erzeugt.

Lösungsvorschlag

$$P = \left\{ \begin{array}{l} Z_0 \rightarrow xZ_1 \mid yZ_3 \mid \varepsilon \\ Z_1 \rightarrow yZ_3 \mid xZ_1 \\ Z_2 \rightarrow xZ_2 \mid x \mid yZ_3 \\ Z_3 \rightarrow zZ_2 \mid z \end{array} \right\}$$

Der Automat auf flaci.com (FLACI: Formale Sprachen, abstrakte Automaten, Compiler und Interpreter) Ein Projekt der Hochschule Zittau/Görlitz und der Pädagogischen Hochschule Schwyz: flaci.com/Gjfc3c2d2

### 66115 / 2020 / Herbst / Thema 1 / Teilaufgabe 1 / Aufgabe 3

Seien  $\Sigma = \{a, b, c\}$  und  $L = \{wc\hat{w} \mid w \in \{a, b\}^*\}$ . Dabei ist  $\hat{w}$  das zu  $w$  gespiegelte Wort.

(a) Zeigen Sie, dass  $L$  nicht regulär ist.

**Exkurs: Pumping-Lemma für Reguläre Sprachen**

Es sei  $L$  eine reguläre Sprache. Dann gibt es eine Zahl  $j$ , sodass für alle Wörter  $\omega \in L$  mit  $|\omega| \geq j$  (jedes Wort  $\omega$  in  $L$  mit Mindestlänge  $j$ ) jeweils eine Zerlegung  $\omega = uvw$  existiert, sodass die folgenden Eigenschaften erfüllt sind:

- (i)  $|v| \geq 1$  (Das Wort  $v$  ist nicht leer.)
- (ii)  $|uv| \leq j$  (Die beiden Wörter  $u$  und  $v$  haben zusammen höchstens die Länge  $j$ .)
- (iii) Für alle  $i = 0, 1, 2, \dots$  gilt  $uv^i w \in L$  (Für jede natürliche Zahl (mit 0)  $i$  ist das Wort  $uv^i w$  in der Sprache  $L$ )

Die kleinste Zahl  $j$ , die diese Eigenschaften erfüllt, wird Pumping-Zahl der Sprache  $L$  genannt.

Lösungsvorschlag

$L$  ist regulär. Dann gilt für  $L$  das Pumping-Lemma. Sei  $j$  die Zahl aus dem Pumping-Lemma. Dann muss sich das Wort  $a^j b c b a^j \in L$  aufpumpen lassen (da  $|a^j b c b a^j| \geq j$ ).  $a^j b c b a^j = uvw$  ist eine passende Zerlegung laut Lemma. Da  $|uv| < j$ , ist  $u = a^x$ ,  $v = a^y$ ,  $w = a^z b c b a^j$ , wobei  $y > 0$  und  $x + y + z = j$ . Aber dann  $uv^0 w = a^{x+z} b c b a^j \notin L$ , da  $x + z < j$ . Widerspruch. <sup>a</sup>

<sup>a</sup><https://userpages.uni-koblenz.de/~sofronie/gti-ss-2015/slides/endliche-automaten6.pdf>

- (b) Zeigen Sie, dass  $L$  kontextfrei ist, indem Sie eine geeignete Grammatik angeben und anschließend begründen, dass diese die Sprache  $L$  erzeugt.

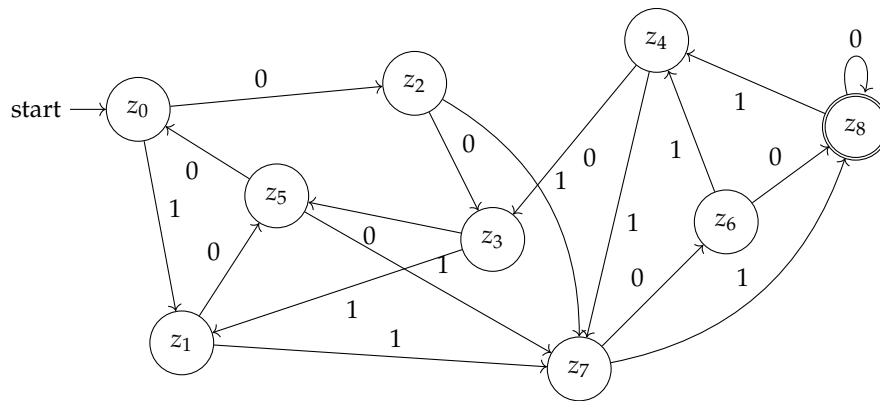
Lösungsvorschlag

$$P = \left\{ \begin{array}{l} S \rightarrow aSa \mid aCa \mid bSb \mid bCb \\ C \rightarrow c \end{array} \right\}$$

$S \vdash aSa \vdash abCba \vdash abcba$   
 $S \vdash bSb \vdash bbSbb \vdash bbaSabb \vdash bbacabb$

**66115 / 2020 / Herbst / Thema 2 / Teilaufgabe 1 / Aufgabe 1**

- (a) Geben Sie einen deterministischen endlichen Automaten (DEA) mit minimaler Anzahl an Zuständen an, der dieselbe Sprache akzeptiert wie folgender deterministischer endlicher Automat. Dokumentieren Sie Ihr Vorgehen geeignet.



Der Automat auf flaci.com (FLACI: Formale Sprachen, abstrakte Automaten, Compiler und Interpreter) Ein Projekt der Hochschule Zittau/Görlitz und der Pädagogischen Hochschule Schwyz: flaci.com/Aj5aei652

Lösungsvorschlag

### Minimierungstabelle (Table filling)

— Der Minimierungs-Algorithmus (auch Table-Filling-Algorithmus genannt) trägt in seinem Verlauf eine Markierung in alle diejenigen Zellen der Tabelle ein, die zueinander nicht äquivalente Zustände bezeichnen. Die Markierung „ $x_n$ “ in einer Tabellenzelle  $(i, j)$  bedeutet dabei, dass das Zustandspaar  $(i, j)$  in der  $k$ -ten Iteration des Algorithmus markiert wurde und die Zustände  $i$  und  $j$  somit zueinander  $(k - 1)$ -äquivalent, aber nicht  $k$ -äquivalent und somit insbesondere nicht äquivalent sind. Bleibt eine Zelle bis zum Ende unmarkiert, sind die entsprechenden Zustände zueinander äquivalent.

$z_0$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$
$z_1$	$x_3$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$
$z_2$	$x_3$	$x_4$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$
$z_3$		$x_3$	$x_3$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$
$z_4$	$x_3$	$x_4$		$x_3$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$
$z_5$	$x_3$	$x_4$		$x_3$		$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$
$z_6$	$x_2$	$x_2$	$x_2$	$x_2$	$x_2$	$x_2$	$\emptyset$	$\emptyset$	$\emptyset$
$z_7$	$x_2$	$x_2$	$x_2$	$x_2$	$x_2$	$x_2$	$x_2$	$\emptyset$	$\emptyset$
$z_8$	$x_1$	$x_1$	$x_1$	$x_1$	$x_1$	$x_1$	$x_1$	$x_1$	$\emptyset$
	$z_0$	$z_1$	$z_2$	$z_3$	$z_4$	$z_5$	$z_6$	$z_7$	$z_8$

$x_1$  Paar aus End-/ Nicht-Endzustand kann nicht äquivalent sein.

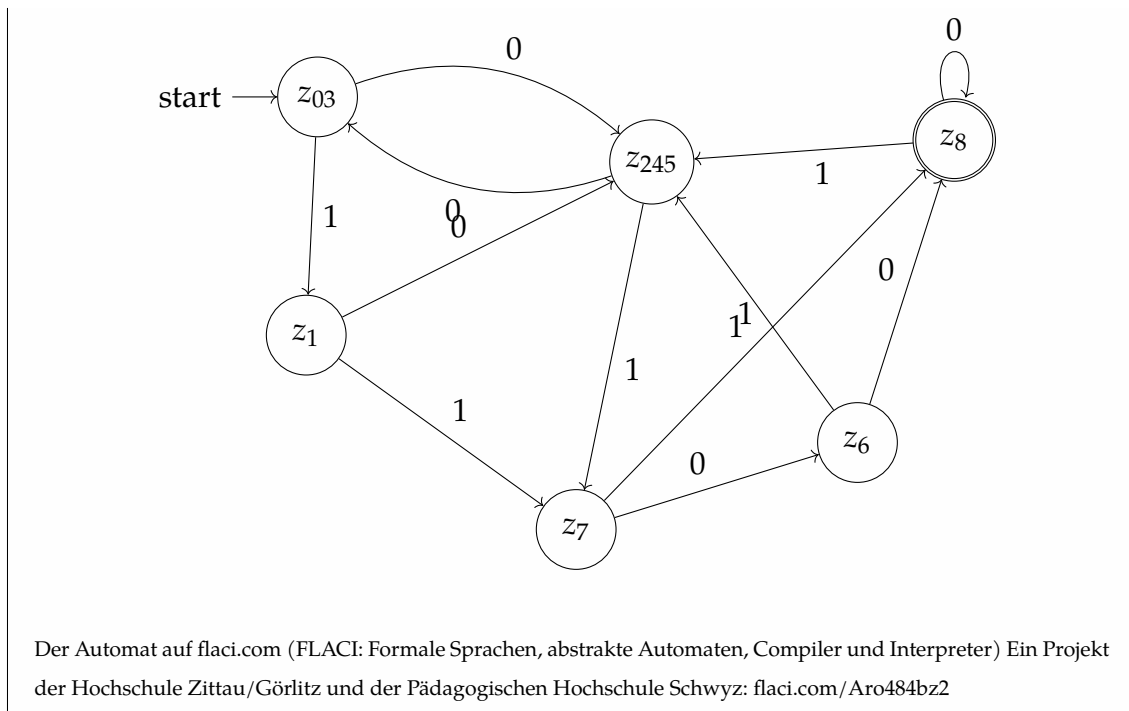
$x_2$  Test, ob man mit der Eingabe zu einem bereits markiertem Paar kommt.

$x_3$  In weiteren Iterationen markierte Zustände.

$x_4$  ...

## Übergangstabelle

Zustandspaar	0	1
$(z_0, z_1)$	$(z_2, z_5)$	$(z_1, z_7) \ x_3 \ x_3$
$(z_0, z_2)$	$(z_2, z_3)$	$(z_1, z_7) \ x_3$
$(z_0, z_3)$	$(z_2, z_5)$	$(z_1, z_1)$
$(z_0, z_4)$	$(z_2, z_3)$	$(z_1, z_7) \ x_3$
$(z_0, z_5)$	$(z_2, z_0)$	$(z_1, z_7) \ x_3$
$(z_0, z_6)$	$(z_2, z_8)$	$(z_1, z_4) \ x_2$
$(z_0, z_7)$	$(z_2, z_6)$	$(z_1, z_8) \ x_2$
$(z_1, z_2)$	$(z_5, z_3)$	$(z_7, z_7) \ x_4$
$(z_1, z_3)$	$(z_5, z_5)$	$(z_7, z_1) \ x_3$
$(z_1, z_4)$	$(z_5, z_3)$	$(z_7, z_7) \ x_4$
$(z_1, z_5)$	$(z_5, z_0)$	$(z_7, z_7) \ x_4$
$(z_1, z_6)$	$(z_5, z_8)$	$(z_7, z_4) \ x_2$
$(z_1, z_7)$	$(z_5, z_6)$	$(z_7, z_8) \ x_2$
$(z_2, z_3)$	$(z_3, z_5)$	$(z_7, z_1) \ x_3$
$(z_2, z_4)$	$(z_3, z_3)$	$(z_7, z_7)$
$(z_2, z_5)$	$(z_3, z_0)$	$(z_7, z_7)$
$(z_2, z_6)$	$(z_3, z_8)$	$(z_7, z_4) \ x_2$
$(z_2, z_7)$	$(z_3, z_6)$	$(z_7, z_8) \ x_2$
$(z_3, z_4)$	$(z_5, z_3)$	$(z_1, z_7) \ x_3$
$(z_3, z_5)$	$(z_5, z_0)$	$(z_1, z_7) \ x_3$
$(z_3, z_6)$	$(z_5, z_8)$	$(z_1, z_4) \ x_2$
$(z_3, z_7)$	$(z_5, z_6)$	$(z_1, z_8) \ x_2$
$(z_4, z_5)$	$(z_3, z_0)$	$(z_7, z_7)$
$(z_4, z_6)$	$(z_3, z_8)$	$(z_7, z_4) \ x_2$
$(z_4, z_7)$	$(z_3, z_6)$	$(z_7, z_8) \ x_2$
$(z_5, z_6)$	$(z_0, z_8)$	$(z_7, z_4) \ x_2$
$(z_5, z_7)$	$(z_0, z_6)$	$(z_7, z_8) \ x_2$
$(z_6, z_7)$	$(z_8, z_6)$	$(z_4, z_8) \ x_2$



- (b) Beweisen oder widerlegen Sie für folgende Sprachen über dem Alphabet  $\Sigma = \{a, b, c\}$ , dass sie regulär sind.

(i)  $L_1 = \{a^i c u b^j v a c^k \mid u, v \in \{a, b\}^* \text{ und } i, j, k \in \mathbb{N}_0\}$

Lösungsvorschlag

Die Sprache  $L_1$  ist regulär. Nachweis durch regulären Ausdruck:

$$a^* c (a|b)^* b^* (a|b)^* a c^*$$

(ii)  $L_2 = \{a^i c u b^j v a c^k \mid u, v \in \{a, b\}^* \text{ und } i, j, k \in \mathbb{N}_0 \text{ mit } k = i + j\}$

Lösungsvorschlag

Die Sprache  $L_2$  ist nicht regulär. Widerlegung durch das Pumping-Lemma.  
TODO

- (c) Sei  $L$  eine reguläre Sprache über dem Alphabet  $\Sigma$ . Für ein festes Element  $a \in \Sigma$  betrachten wir die Sprache  $L_a = \{aw \mid w \in \Sigma^*, wa \in L\}$ . Zeigen Sie, dass  $L_a$  regulär ist.

Lösungsvorschlag

Die regulären Sprachen sind unter dem Komplement abgeschlossen.

## 66115 / 2021 / Frühjahr / Thema 1 / Teilaufgabe 1 / Aufgabe 1

Im Folgenden bezeichnet  $a^i = a \dots a$  und  $\varepsilon$  steht für das leere Wort (öinsbesondere  $a^0 = \varepsilon$ ).

Die Menge  $\mathbb{N}_0 = \{0, 1, 2, \dots\}$  ist die Menge aller nicht-negativer Ganzzahlen.  
Die Sprachen  $L_1, \dots, L_{12}$  seien definiert als:

- (a) Ordnen Sie jedem der folgenden nichtdeterministischen endlichen Automaten  $N_j, j = 1, \dots, 6$ , (die alle über dem Alphabet  $\Sigma = \{a\}$  arbeiten) **jeweils eine** der Sprachen  $L_i \in \{L_1, \dots, L_{12}\}$  zu, sodass  $L_i$ , genau die von  $N_i$ , **akzeptierte Sprache** ist.

Lösungsvorschlag

- $N_1 = L_6$  (mindestens ein  $a$ )
- $N_2 = L_8$  (ungerade Anzahl an  $a$ 's:  $1, 5, 7, \dots$ )
- $N_3 = L_2$  (gerade Anzahl an  $a$ 's:  $2, 4, 6, \dots$ )
- $N_4 = L_{12}$  (leeres Wort)
- $N_5 = L_8$  (ungerade Anzahl an  $a$ 's:  $1, 5, 7, \dots$ )
- $N_6 = L_{11}$  (die Sprache akzeptiert nicht)

- (b) Zeigen Sie für eine der Sprachen  $L_1, \dots, L_{12}$  dass diese **nicht regulär** ist.

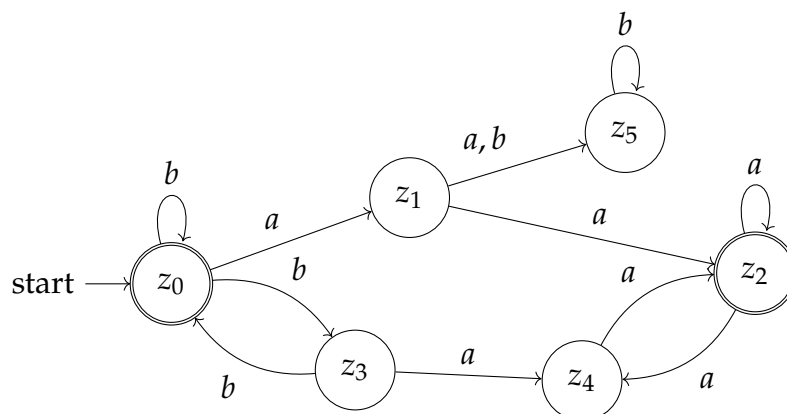
Lösungsvorschlag

$$L_1 0 = \{a^n \mid n \in \mathbb{N}_0, n \text{ ist Primzahl}\}$$

ist nicht regulär, da sich sonst jede Primzahl  $p$  einer bestimmten Mindestgröße  $j$  als Summe von natürlichen Zahlen  $u + v + w$  darstellen ließe, so dass  $v \geq 1$  und für alle  $i \geq 0$  auch  $u + iv + w = p + (i1)v$  prim ist. Dies ist jedoch für  $i = p + 1$  wegen  $p + (p + 11)v = p(1 + v)$  nicht der Fall.<sup>a</sup>

<sup>a</sup><https://www.informatik.hu-berlin.de/de/forschung/gebiete/algorithmenII/Lehre/ws13/einftheo/einftheo-skript.pdf>

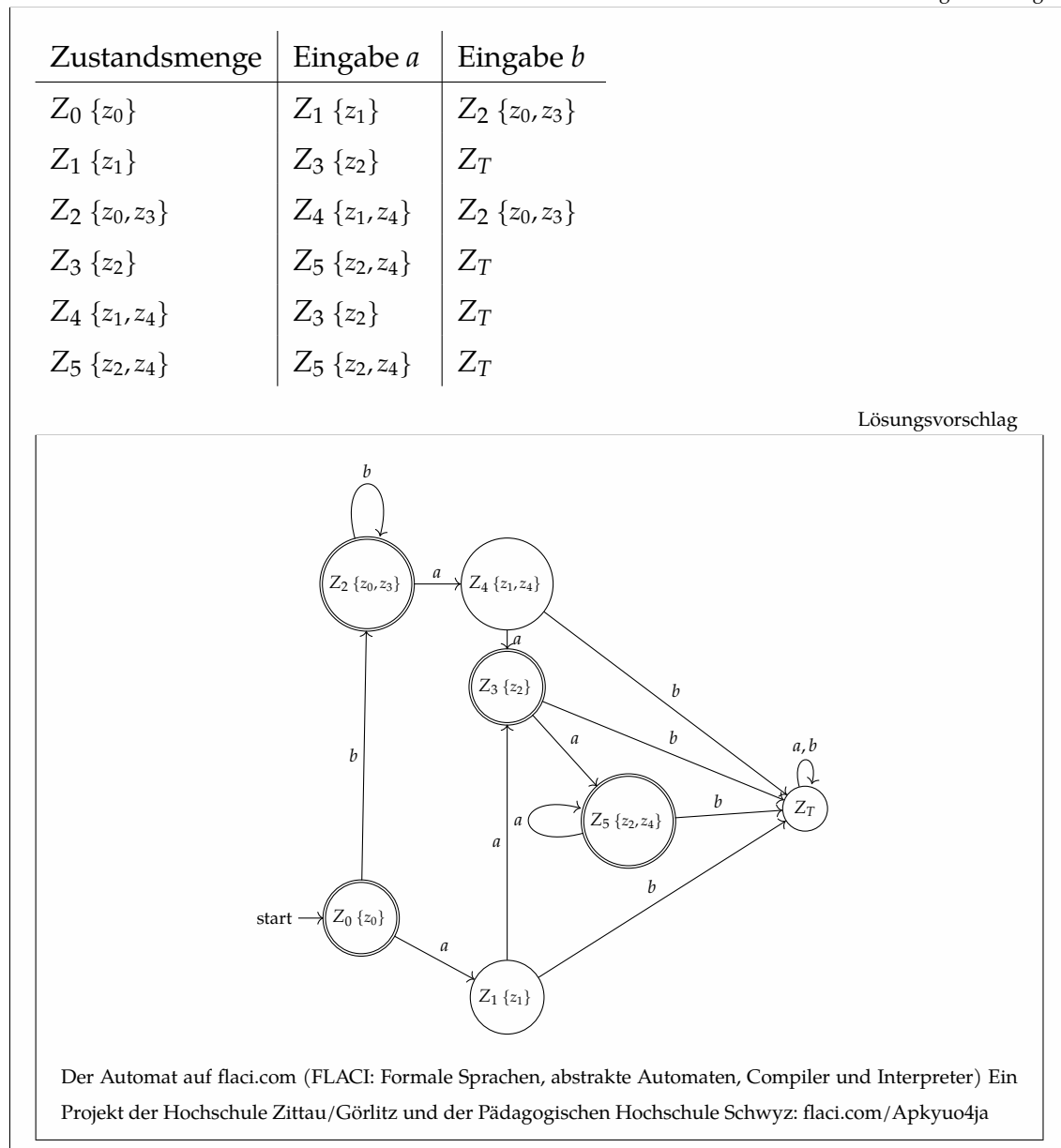
- (c) Konstruieren Sie für den folgenden nichtdeterministischen endlichen Automaten (der Worte über dem Alphabet  $\Sigma = \{a, b\}$  verarbeitet) einen äquivalenten deterministischen endlichen Automaten mithilfe der Potenzmengenkonstruktion. Zeichnen Sie dabei nur die vom Startzustand erreichbaren Zustände. Erläutern Sie Ihr Vorgehen.





Der Automat auf flaci.com (FLACI: Formale Sprachen, abstrakte Automaten, Compiler und Interpreter) Ein Projekt der Hochschule Zittau/Görlitz und der Pädagogischen Hochschule Schwyz: flaci.com/Af7iooyca

Lösungsvorschlag



Lösungsvorschlag

## 66115 / 2021 / Frühjahr / Thema 2 / Teilaufgabe 1 / Aufgabe 1

(a) Sei

$L_1 = \{w \in \{a, b, c\}^* \mid w \text{ enthält genau zweimal den Buchstaben } a \text{ und der vorletzte Buchstabe ist } c\}$ .  
Geben Sie einen regulären Ausdruck für die Sprache  $L_1$  an.

Lösungsvorschlag

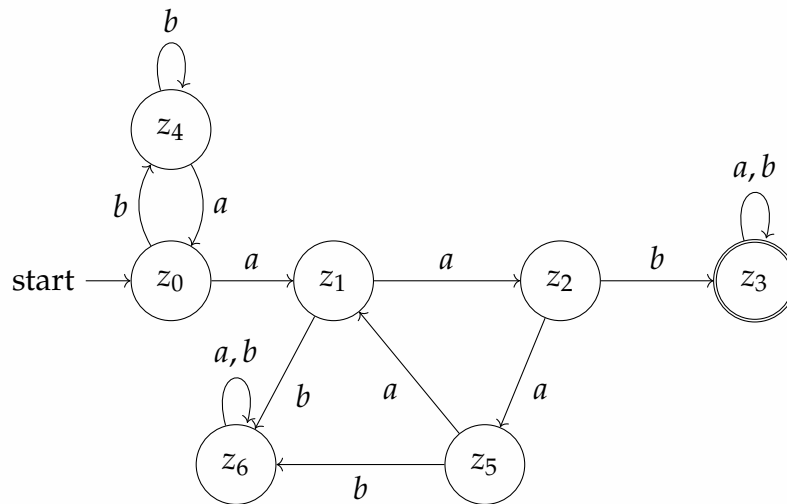
```
(
  ((b|c)* a (b|c)* a (b|c)* c (b|c))
  |
```

$((b|c)^* a (b|c)^* c a)$   
)

- (b) Konstruieren Sie einen deterministischen endlichen Automaten für die Sprache  $L_2$ :

$$L_2 = \{ w \in \{a, b\}^* \mid w \text{ enthält genau einmal das Teilwort } aab \}$$

Lösungsvorschlag



Der Automat auf flaci.com (FLACI: Formale Sprachen, abstrakte Automaten, Compiler und Interpreter) Ein Projekt der Hochschule Zittau/Görlitz und der Pädagogischen Hochschule Schwyz: flaci.com/Ahf2oduri

- (c) Sei  $\mathbb{N} = \{1, 2, 3, \dots\}$  die Menge der strikt positiven natürlichen Zahlen. Sei  $L_3 = \{ \#a^{i_1}\#a^{i_2}\#\dots\#a^{i_{n-1}}\#a^{i_n}\# \mid n, i_1, \dots, i_n \in \mathbb{N} \text{ und es existiert } j \in \mathbb{N} \text{ mit } i_j = n + 1 \}$  eine Sprache über Alphabet  $\{\#, a\}$ .  
So ist z. B.  $\#a\#aaa\# \in L_3$  (da das Teilwort  $a^3 = aaa$  vorkommt) und  $\#a\#a\#a\#a\# \in L_3$  (da das Teilwort  $a^5 = aaaaa$  nicht vorkommt). Beweisen Sie, dass  $L_3$  nicht regulär ist.

# Kontextfreie Sprache

66115 / 2021 / Frühjahr / Thema 2 / Aufgabe 1

## Ableitungen

Bestimmen Sie für die folgende Grammatik jeweils für die angegebenen Wörter  $w_i$  mehrere Ableitungen sowie Parsebäume, wenn dies nicht eindeutig möglich ist.

$P = \{$

$$S \rightarrow AB$$

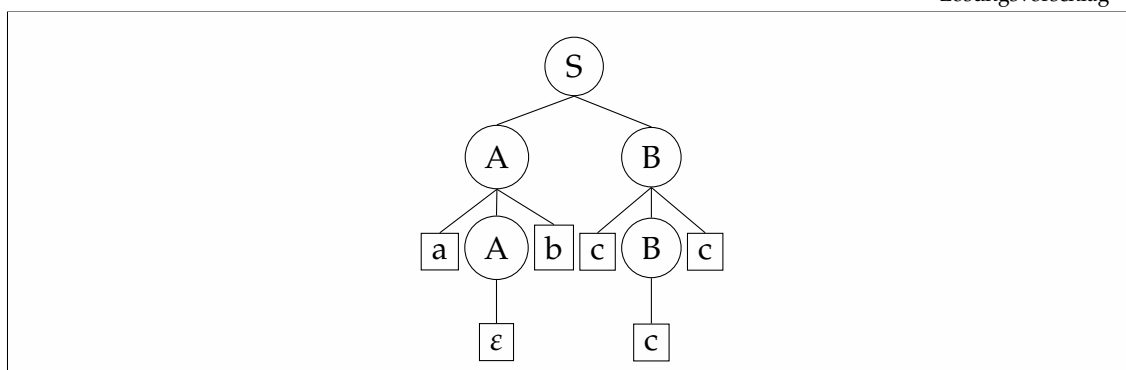
$$A \rightarrow aAb \mid \epsilon$$

$$B \rightarrow cBc \mid c$$

$\}$

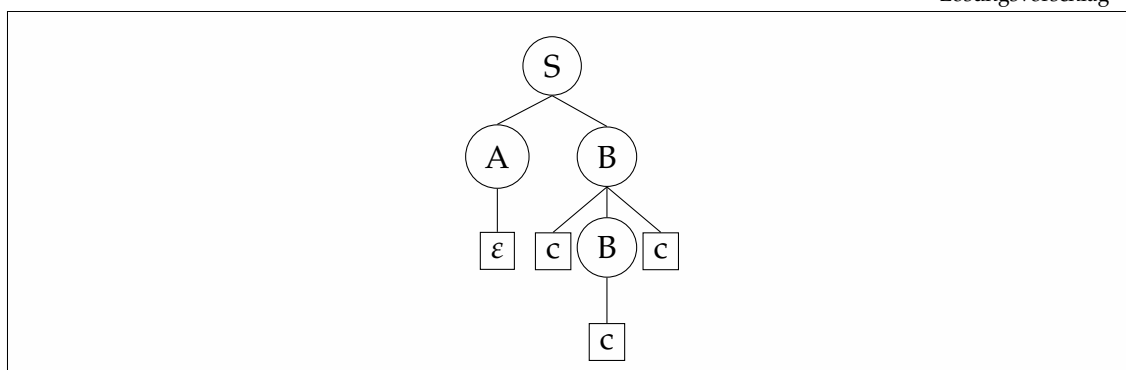
(a)  $w_1 = abccc$

Lösungsvorschlag

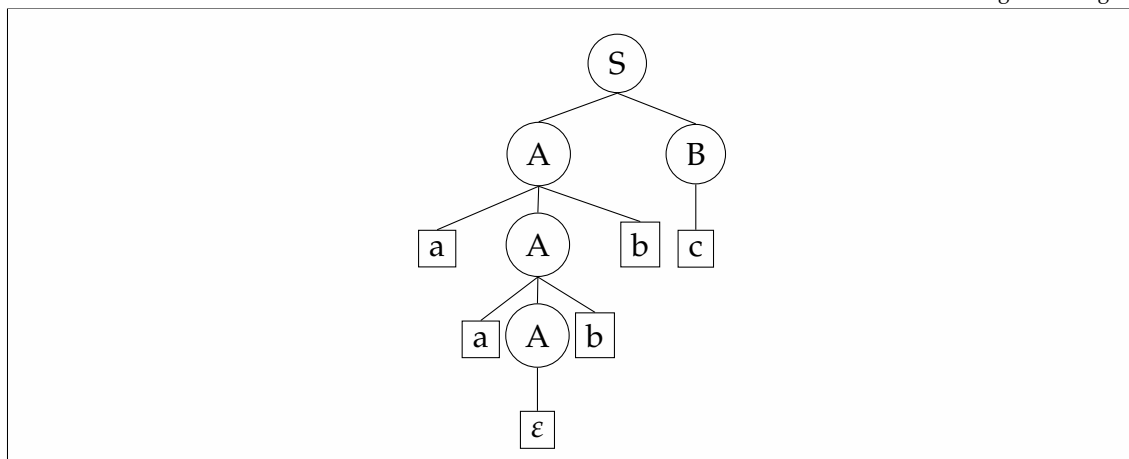


(b)  $w_2 = ccc$

Lösungsvorschlag



(c)  $w_3 = aabbc$



Welche Sprache wird durch die Grammatik erzeugt?

Lösungsvorschlag

Die Sprache beinhaltet alle Wörter, die gleich viele  $a$ 's gefolgt von gleich vielen  $b$ 's und auf ungeradzahlig viele  $c$ 's endet.

## 66115 / 2021 / Frühjahr / Thema 2 / Aufgabe 1

### Ableitung und Parsebaum

- (a) Erstelle eine Ableitung und einen Parsebaum für die folgende Grammatik für das Wort

Der Automat auf flaci.com (FLACI: Formale Sprachen, abstrakte Automaten, Compiler und Interpreter) Ein Projekt der Hochschule Zittau/Görlitz und der Pädagogischen Hochschule Schwyz: flaci.com/Gf6s60uxg

$$G = (\{P\}, \{0, 1\}, P, S)$$

$$P = \{$$

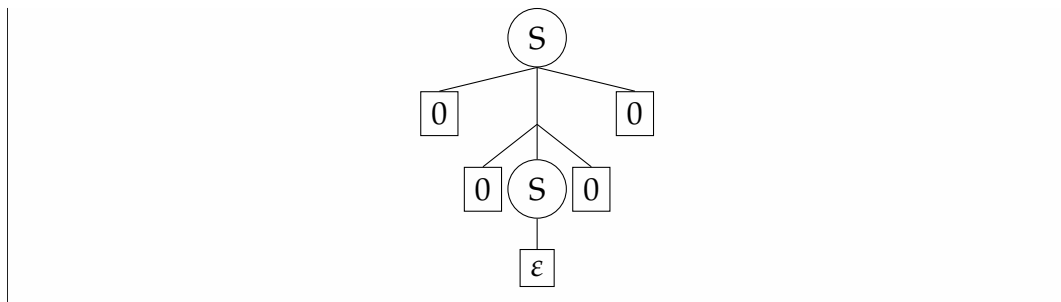
$$S \rightarrow \varepsilon \mid 0 \mid 1 \mid 0S0 \mid 1S1$$

$$\}$$

- 0000

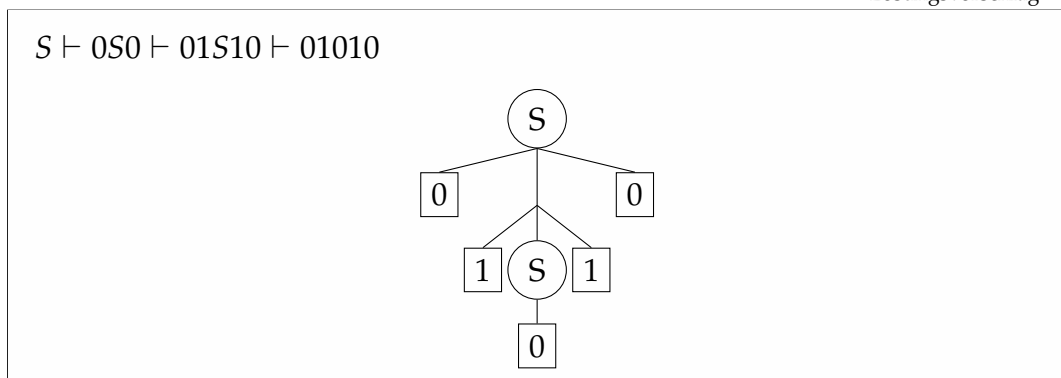
Lösungsvorschlag

$$S \vdash 0S0 \vdash 00S00 \vdash 0000$$



- 01010

Lösungsvorschlag



(b) Erstelle eine Ableitung und einen Parsebaum für die nebenstehende Grammatik für das Wort

Der Automat auf flaci.com (FLACI: Formale Sprachen, abstrakte Automaten, Compiler und Interpreter) Ein Projekt der Hochschule Zittau/Görlitz und der Pädagogischen Hochschule Schwyz: flaci.com/Gpmohr81a

$$V = \{S, A, B\}$$

$$\Sigma = \{0, 1\}$$

$$P = \left\{ \right.$$

$$S \rightarrow A1B$$

$$A \rightarrow 0A \mid \varepsilon$$

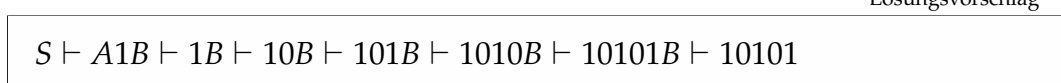
$$B \rightarrow 0B \mid 1B \mid \varepsilon$$

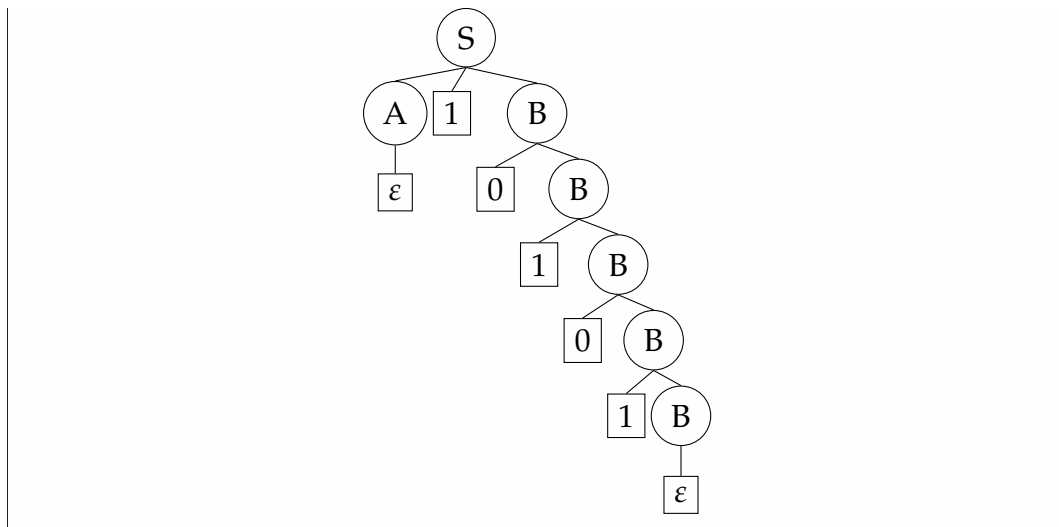
$\left. \right\}$

$$S = S$$

- 10101

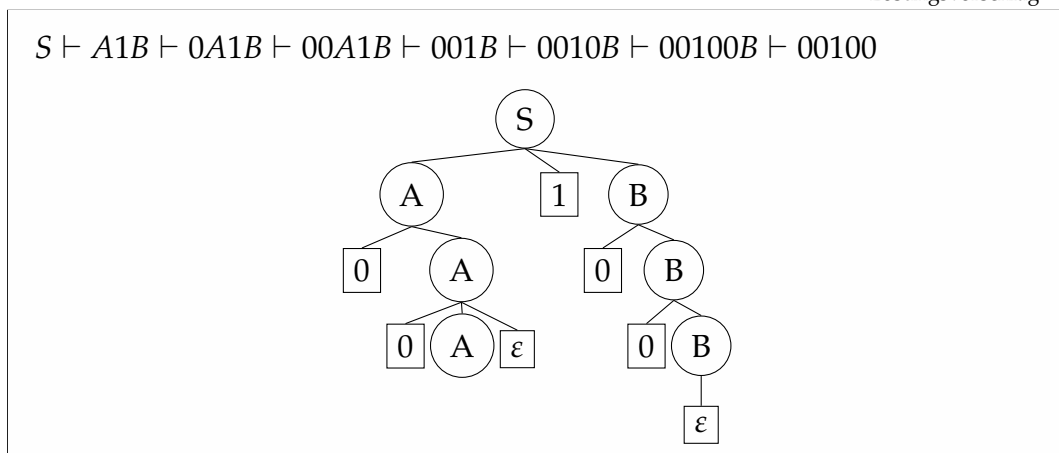
Lösungsvorschlag





- 00100

Lösungsvorschlag



(c) Sind die Parsebäume eindeutig?

Lösungsvorschlag

Ja, die Parsebäume sind eindeutig.

## 66115 / 2021 / Frühjahr / Thema 2 / Aufgabe 1

### Klammerausdrücke

In Programmierumgebungen kommen Abfolgen von Klammern, runde, eckige und geschweifte, vor. Diese müssen in der richtigen Abfolge auf- bzw. geschlossen werden. Eine korrekte Abfolge von Klammern wäre zum Beispiel:

{ [ ( ) { ( ) { [ ( [ ] ) } } } }

(a) Entwerfen Sie eine Grammatik, die die korrekte Abfolge solcher Klammerfolgen beschreibt.

$$S \rightarrow \{ S \} \mid ( S ) \mid [ S ] \mid S S \mid \text{EPSILON}$$

Der Automat auf flaci.com (FLACI: Formale Sprachen, abstrakte Automaten, Compiler und Interpreter) Ein Projekt der Hochschule Zittau/Görlitz und der Pädagogischen Hochschule Schwyz: flaci.com/Ghjeb39xr

(b) Geben Sie eine Ableitung für den oben angegebenen Klammerausdruck an.

Der oben angegebene Klammerausdruck mit Einrückungen

```

01 {
02   []
03   (
04   {
05     (
06     {
07       []
08       (
09         []
10       )
11     }
12   }
13 }
14 {}

```

Die Ableitung:

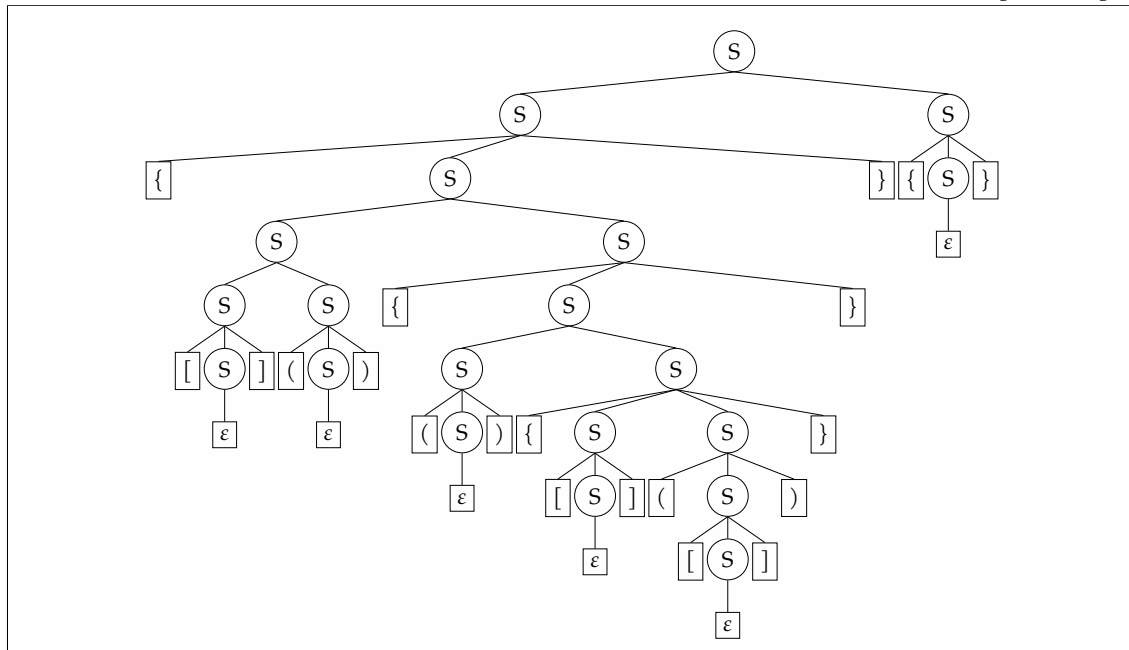
```

S ->
SS ->
{S}S ->           siehe Zeile 01 13
{SS}S ->
{[S]S}S ->        siehe Zeile 02
{[]S}S ->
{[]SS}S ->
{[] (S)S}S ->     siehe Zeile 03
{[] () {S}}S ->   siehe Zeile 04 12
{[] () {SS}}S ->
{[] () {(S)S}}S ->
{[] () {( )S}}S -> siehe Zeile 05
{[] () {( ) {S}}S -> siehe Zeile 06 11
{[] () {( ) {SS}}S ->
{[] () {( ) {[S]S}}S -> siehe Zeile 07
{[] () {( ) {[]S}}S ->
{[] () {( ) {[] (S)}}S -> siehe Zeile 08 10
{[] () {( ) {[] ([S])}}S -> siehe Zeile 09
{[] () {( ) {[] ([[]])}}S ->
{[] () {( ) {[] ([[]])}}{S} -> siehe Zeile 14
{[] () {( ) {[] ([[]])}}{}

```

(c) Zeichnen Sie eine Ableitungsbaum für den oben angegebenen Klammerausdruck.

## Lösungsvorschlag



66115 / 2021 / Frühjahr / Thema 2 / Aufgabe 1

# Kontextfreie Grammatiken

Geben Sie für die folgenden Sprachen eine kontextfreie Grammatik an:

- Alle Wörter der Sprachen bestehen aus  $a$ 's, gefolgt von gleich vielen  $b$ 's.

Lösungsvorschlag

$$P = \left\{ \begin{array}{l} S \rightarrow aSb \mid \varepsilon \end{array} \right.$$

- Die Wörter der Sprache bestehen aus gleich vielen  $x$  wie  $y$ .

Lösungsvorschlag

$$\mathbf{P} = \left\{ \begin{array}{l} S \rightarrow xSY \mid ySX \mid \varepsilon \\ X \rightarrow xS \\ Y \rightarrow yS \end{array} \right.$$



## 66115 / 2021 / Frühjahr / Thema 2 / Aufgabe 1

## Kontextfreie Sprache

Gegeben ist die Grammatik  $G = (\{S, A, B\}, \{a, b\}, P, S)$  und den Produktionen

$$P = \left\{ \begin{array}{l} S \rightarrow SAB \mid \varepsilon \\ BA \rightarrow AB \\ AA \rightarrow aa \\ BB \rightarrow bb \end{array} \right\}$$

(a) Geben Sie einen Ausdruck an, der die Wörter der Sprache beschreibt.

Lösungsvorschlag

$$L = \{(a^n b^n)^m \mid m \in \mathbb{N}_0 \text{ und } n \in \text{gerade Zahlen}\}$$

Einige Testableitungen um die Grammatik in Erfahrung zu bringen:

„.“ nur als optische Stütze nach 4 Zeichen eingefügt.

#### Mit 4 Buchstaben

$$S \vdash SAB \vdash SABAB \vdash ABAB \vdash AABB \vdash aabb$$

#### Mit 6 Buchstaben

$$S \vdash \dots \vdash ABAB.AB \vdash AABB.AB \vdash AABA.BB \vdash AAAB.BB \vdash \emptyset$$

#### Mit 8 Buchstaben

$$S \vdash \dots \vdash ABAB.ABAB \vdash \dots \vdash aabb.aabb$$

$$S \vdash \dots \vdash ABAB.ABAB \vdash \dots \vdash AABB.AABB \vdash AABA.BABB \vdash AABA.ABBB \vdash AAAB.ABBB \vdash AAAA.BBBB \vdash aaaa.bbbb$$

#### Mit 12 Buchstaben

$$S \vdash \dots \vdash ABAB.ABAB.ABAB \vdash \dots \vdash aabb.aabb.aabb$$

$$S \vdash \dots \vdash ABAB.ABAB.ABAB \vdash AAAA.BBBB.AABB \vdash aaaa.bbbb.aabb$$

$$S \vdash \dots \vdash ABAB.ABAB.ABAB \vdash AABB.ABAB.ABAB \vdash AABA.BBAB.ABAB \vdash AAAB.BBAB.ABAB \vdash \dots \vdash aaaa.aabb.bbbb$$

(b) Geben Sie eine kontextfreie Grammatik  $G'$  an, für die gilt:  $L(G') = L(G)$

$$P = \left\{ \begin{array}{c} S \rightarrow aaSbb \mid SS \mid \varepsilon \end{array} \right\}$$

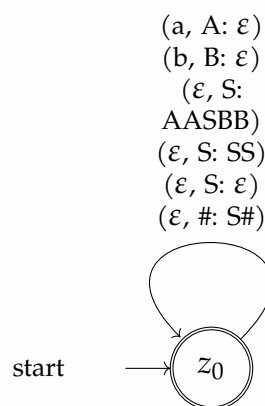
Der Automat auf [flaci.com](http://flaci.com) (FLACI: Formale Sprachen, abstrakte Automaten, Compiler und Interpreter) Ein Projekt der Hochschule Zittau/Görlitz und der Pädagogischen Hochschule Schwyz: [flaci.com/Grn19rt8w](http://flaci.com/Grn19rt8w)

(c) Geben Sie einen Kellerautomaten an, der die Sprache akzeptiert.

Lösungsvorschlag

### 1. Kellerautomat (aus der Grammtik abgeleitet)

$$K = (\{z_0\}, \{a, b\}, \{\#, S, A, B\}, \delta, z_0, \#, \{z_0\})$$

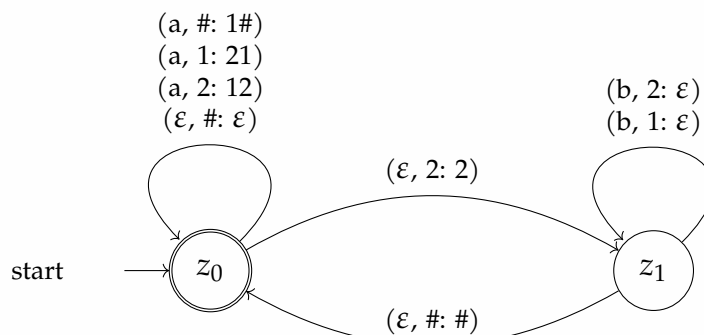


Der Automat auf flaci.com (FLACI: Formale Sprachen, abstrakte Automaten, Compiler und Interpreter) Ein Projekt der Hochschule Zittau/Görlitz und der Pädagogischen Hochschule Schwyz: [flaci.com/Araj960s2](http://flaci.com/Araj960s2)

### 2. Kellerautomat

$$K = (\{z_0, z_1\}, \{a, b\}, \{\#, 1, 2\}, \delta, z_0, \#, \{z_0\})$$

Bemerkung zum Kelleralphabet: 1 steht für 1A, also ein  $a$  befindet sich im Keller, und 2 steht für 2A, also zwei  $a$  befinden sich im Keller.



Der Automat auf flaci.com (FLACI: Formale Sprachen, abstrakte Automaten, Compiler und Interpreter) Ein Projekt der Hochschule Zittau/Görlitz und der Pädagogischen Hochschule Schwyz: [flaci.com/Ahfqseouz](http://flaci.com/Ahfqseouz)

## 66115 / 2021 / Frühjahr / Thema 2 / Aufgabe 1

## Kontextfreie Sprache

## Übung

- (a) Erstelle eine (deterministische) Grammatik für Palindrome, für die ein DPDA existiert.

$$L = \{ w\$w^R \mid w \in (a|b)^* \}$$

- (b) Wandle diese Grammatik in einen DPDA um.

## Übung

Überführe die folgenden kontextfreien Grammatiken in CNF

$$P = \left\{ \begin{array}{l} S \rightarrow ABC \\ A \rightarrow aCD \\ B \rightarrow bCD \\ C \rightarrow D \mid \varepsilon \\ D \rightarrow C \end{array} \right\}$$

## 66115 / 2021 / Frühjahr / Thema 2 / Aufgabe 1

Teste jeweils mit dem CYK-Algorithmus, ob das angegebene Wort zur Sprache der Grammatik gehört.

$$P = \left\{ \begin{array}{l} S \rightarrow SS \mid R_a A \mid R_b B \mid R_c C \\ A \rightarrow R_b R_c \mid R_c R_b \\ B \rightarrow R_a R_c \mid R_c R_a \\ C \rightarrow R_a R_b \mid R_b R_a \\ R_a \rightarrow a \\ R_b \rightarrow b \\ R_c \rightarrow c \end{array} \right\}$$

- (a)  $\omega_1 = acbcab$

a	c	b	c	a	b
$R_a$	$R_c$	$R_b$	$R_c$	$R_a$	$R_b$
B	A	A	B	C	
S	-	S	S		
-	-	-			
-	-				
S					

$\Rightarrow \omega_1 \in L(G)$

(b)  $\omega_2 = cabb$

c	a	b	b
$R_c$	$R_a$	$R_b$	$R_b$
B	C	-	
S	-		
-			

$\Rightarrow \omega_2 \notin L(G)$

## 66115 / 2021 / Frühjahr / Thema 2 / Aufgabe 1

### Wortproblem - CYK-Algorithmus

Teste jeweils mit dem CYK-Algorithmus, ob das angegebene Wort zur Sprache der Grammatik gehört.

(a) Grammatik  $G_1$ :

$$P = \left\{ \right.$$

$$\begin{aligned}
 S &\rightarrow SS \mid R_a A \mid R_b B \mid R_c C \\
 A &\rightarrow R_b R_c \mid R_c R_b \\
 B &\rightarrow R_a R_c \mid R_c R_a \\
 C &\rightarrow R_a R_b \mid R_b R_a \\
 R_a &\rightarrow a \\
 R_b &\rightarrow b \\
 R_c &\rightarrow c
 \end{aligned}$$

}

(i)  $w_1 = abccab$

(ii)  $w_2 = abcba$

(b) Grammatik  $G_2$ :

$$P = \{$$

$$\begin{aligned}
 Z &\rightarrow XD \mid XA \mid \varepsilon \\
 S &\rightarrow XD \mid XA \\
 A &\rightarrow V_a D \mid V_a A \mid AB \mid c \\
 B &\rightarrow BB \mid CC \mid c \mid a \\
 C &\rightarrow CC \mid c \\
 D &\rightarrow AB \\
 X &\rightarrow V_a D \mid V_a A \mid AB \mid b \mid c \\
 V_a &\rightarrow a
 \end{aligned}$$

}

(i)  $w_3 = bacac$

Lösungsvorschlag

b	a	c	a	c
X	$B, V_a$	$A, B, C, X$	$B, V_a$	$A, B, C, X$
-	$B, A, X$	$A, D, X, B$	$B, A, X$	
$Z, S$	$A, B, D, X$	$B, A, X, D, S, Z$		
$Z, S$	$A, B, D, S, X, Z$			
$Z, S$				

(ii)  $w_4 = baca$

## 66115 / 2021 / Frühjahr / Thema 2 / Aufgabe 1

## CYK-Algorithmus

 $P = \{$  $S \rightarrow AB \mid BT$  $A \rightarrow BA \mid a$  $B \rightarrow TT \mid b$  $T \rightarrow AB \mid a$  $\}$ 

Lösungsvorschlag

b	a	a	a	b
B	A,T	A,T	A,T	B
A,S	B	B	S,T	
-	S,T,A	B		
S,A,T	S,T			
S,T				

 $\Rightarrow baaab \in L(G)$ 

## 66115 / 2021 / Frühjahr / Thema 2 / Aufgabe 1

CYK-Algorithmus<sup>1</sup> $G = (\{S, A, B, C\}, \{a, b\}, P, S)$  $P = \{$  $S \rightarrow AB \mid BC$  $A \rightarrow BA \mid a$  $B \rightarrow CC \mid b$  $C \rightarrow AB \mid a$  $\}$ <sup>1</sup><https://www.youtube.com/watch?v=Q5TvCyu4RUo>

a	b	a	a	b
A,C	B	A,C	A,C	B
S,C	A,S	B	S,C	
B	-	B		
S,A	-			
S,S,C				

$\Rightarrow abaab \in L(G)$

## 66115 / 2021 / Frühjahr / Thema 2 / Aufgabe 1

### Chomsky-Normalform

Überführen Sie jeweils die angegebene kontextfreie Grammatik in Chomsky-Normalform.

(a)  $G = (\{S, A, B, C, X\}, \{a, b, c\}, P, S)$  mit  $P$ :

$$P = \left\{ \begin{array}{l} S \rightarrow XAB \mid \varepsilon \\ A \rightarrow aAB \mid AB \mid c \\ B \rightarrow BB \mid C \mid a \\ C \rightarrow CC \mid c \mid \varepsilon \\ X \rightarrow A \mid b \end{array} \right\}$$

(b)  $G = (\{S, T\}, \{a, b, c\}, P, S)$  mit  $P$ :

$$P = \left\{ \begin{array}{l} S \rightarrow aSbS \mid T \\ T \rightarrow cT \mid c \end{array} \right\}$$

#### (i) Elimination der $\varepsilon$ -Regeln

— Alle Regeln der Form  $A \rightarrow \varepsilon$  werden eliminiert. Die Ersetzung von  $A$  wird durch  $\varepsilon$  in allen anderen Regeln vorweggenommen.

∅ Nichts zu tun

#### (ii) Elimination von Kettenregeln



— Jede Produktion der Form  $A \rightarrow B$  mit  $A, B \in S$  wird als Kettenregel bezeichnet. Diese tragen nicht zur Produktion von Terminalzeichen bei und lassen sich ebenfalls eliminieren. —

$$P = \left\{ \begin{array}{l} S \rightarrow aSbS \mid cT \mid c \\ T \rightarrow cT \mid c \end{array} \right\}$$

(iii) **Separation von Terminalzeichen**

— Jedes Terminalzeichen  $\sigma$ , das in Kombination mit anderen Symbolen auftaucht, wird durch ein neues Nonterminal  $S_\sigma$  ersetzt und die Menge der Produktionen durch die Regel  $S_\sigma \rightarrow \sigma$  ergänzt. —

$$P = \left\{ \begin{array}{l} S \rightarrow ASAS \mid CT \mid c \\ T \rightarrow CT \mid c \\ A \rightarrow a \\ B \rightarrow b \\ C \rightarrow c \end{array} \right\}$$

(iv) **Elimination von mehrelementigen Nonterminalketten**

— Alle Produktionen der Form  $A \rightarrow B_1B_2 \dots B_n$  werden in die Produktionen  $A \rightarrow A_{n-1}B_n, A_{n-1} \rightarrow A_{n-2}B_{n-1}, \dots, A_2 \rightarrow B_1B_2$  zerteilt. Nach der Ersetzung sind alle längeren Nonterminalketten vollständig heruntergebrochen und die Chomsky-Normalform erreicht. —

$$P = \left\{ \begin{array}{l} S \rightarrow AU \mid CT \mid c \\ T \rightarrow CT \mid c \\ A \rightarrow a \\ B \rightarrow b \\ C \rightarrow c \\ U \rightarrow SVV \qquad \qquad \qquad \rightarrow AS \end{array} \right\}$$

(c)  $G = (\{S, A, B\}, \{a, b, c\}, P, S)$  mit  $P$ :

$$P = \left\{ \right.$$

$$\begin{aligned}
 S &\rightarrow AB \\
 A &\rightarrow aAA \mid \varepsilon \\
 B &\rightarrow bBB \mid \varepsilon
 \end{aligned}$$

}

## 66115 / 2021 / Frühjahr / Thema 2 / Aufgabe 1

## Kontextfreien Grammatiken in CNF

Überführen Sie die folgenden kontextfreien Grammatiken in CNF (Chomsky-Normalform).

(a)  $P = \left\{ \begin{aligned} S &\rightarrow 0S1 \mid \varepsilon \end{aligned} \right\}$

Lösungsvorschlag

(i) **Elimination der  $\varepsilon$ -Regeln**

— Alle Regeln der Form  $A \rightarrow \varepsilon$  werden eliminiert. Die Ersetzung von  $A$  wird durch  $\varepsilon$  in allen anderen Regeln vorweggenommen.

$$P = \left\{ \begin{aligned} S &\rightarrow 0S1 \mid 01 \end{aligned} \right\}$$

Der Automat auf flaci.com (FLACI: Formale Sprachen, abstrakte Automaten, Compiler und Interpreter) Ein Projekt der Hochschule Zittau/Görlitz und der Pädagogischen Hochschule Schwyz: flaci.com/Ghje1ygz9

(ii) **Elimination von Kettenregeln**

— Jede Produktion der Form  $A \rightarrow B$  mit  $A, B \in S$  wird als Kettenregel bezeichnet. Diese tragen nicht zur Produktion von Terminalzeichen bei und lassen sich ebenfalls eliminieren.

Ø Nichts zu tun

(iii) **Separation von Terminalzeichen**

— Jedes Terminalzeichen  $\sigma$ , das in Kombination mit anderen Symbolen auftaucht, wird durch ein neues Nonterminal  $S_\sigma$  ersetzt und die Menge der Produktionen durch die Regel  $S_\sigma \rightarrow \sigma$  ergänzt.

N = Null, E = Eins

$$P = \left\{ \begin{aligned} S &\rightarrow NSE \mid NE \\ N &\rightarrow 0 \\ E &\rightarrow 1 \end{aligned} \right.$$

(iv) **Elimination von mehrelementigen Nonterminalketten**

— Alle Produktionen der Form  $A \rightarrow B_1 B_2 \dots B_n$  werden in die Produktionen  $A \rightarrow A_{n-1} B_n, A_{n-1} \rightarrow A_{n-2} B_{n-1}, \dots, A_2 \rightarrow B_1 B_2$  zerteilt. Nach der Ersetzung sind alle längeren Nonterminalketten vollständig heruntergebrochen und die Chomsky-Normalform erreicht. —

$$P = \{$$

$$S \rightarrow NR \mid NE$$

$$R \rightarrow SE$$

$$N \rightarrow 0$$

$$E \rightarrow 1$$

$$\}$$

(b)  $P = \{$

$$S \rightarrow a \mid aA \mid B$$

$$A \rightarrow aBB \mid \varepsilon$$

$$B \rightarrow Aa \mid b$$

$$\}$$

Der Automat auf flaci.com (FLACI: Formale Sprachen, abstrakte Automaten, Compiler und Interpreter) Ein Projekt der Hochschule Zittau/Görlitz und der Pädagogischen Hochschule Schwyz: flaci.com/G54gubr9i

**(i) Elimination der  $\varepsilon$ -Regeln**

— Alle Regeln der Form  $A \rightarrow \varepsilon$  werden eliminiert. Die Ersetzung von  $A$  wird durch  $\varepsilon$  in allen anderen Regeln vorweggenommen.

$$P = \left\{ \begin{array}{l} S \rightarrow a \mid aA \mid B \\ A \rightarrow aBB \\ B \rightarrow Aa \mid b \mid a \end{array} \right\}$$

Das leere Wort ist nicht in der Sprache ( $\varepsilon \notin L(G)$ ). In der Sprache sind immer Wörter mit mindestens einem Buchstaben. In der ersten Produktionsregel wird aus  $aA \rightarrow a\varepsilon$  nur das  $a$ . Das ist aber bereits in der ersten Regel enthalten. In der zweiten Regel wird das leere Wort weg gelassen. In der dritten Regel wird noch ein  $a$  hinzugefügt, das aus  $Aa \rightarrow \varepsilon a \rightarrow a$  entstanden ist.

**(ii) Elimination von Kettenregeln**

— Jede Produktion der Form  $A \rightarrow B$  mit  $A, B \in S$  wird als Kettenregel bezeichnet. Diese tragen nicht zur Produktion von Terminalzeichen bei und lassen sich ebenfalls eliminieren. —

$$P = \left\{ \begin{array}{l} S \rightarrow a \mid aA \mid Aa \mid b \\ A \rightarrow aBB \\ B \rightarrow Aa \mid b \mid a \end{array} \right\}$$

Wir schreiben die Regel, die keine einzelnes Nonterminal auf der rechten Seite enthalten, ab. In der ersten Regel wird  $B$  mit  $Aa|b|a$  ersetzt, wobei das letzte  $a$ , dann weggelassen werden kann, da es bereits am Anfang der rechten Seite vorkommt. Die  $B$ -Regel kann nicht weggelassen werden, weil sie in der  $A$ -Regel vorkommt.

(iii) **Separation von Terminalzeichen**

— Jedes Terminalzeichen  $\sigma$ , das in Kombination mit anderen Symbolen auftaucht, wird durch ein neues Nonterminal  $S_\sigma$  ersetzt und die Menge der Produktionen durch die Regel  $S_\sigma \rightarrow \sigma$  ergänzt. —

$$P = \left\{ \begin{array}{l} S \rightarrow a \mid VA \mid AV \mid b \\ A \rightarrow VBB \\ B \rightarrow AV \mid b \mid a \\ V \rightarrow a \end{array} \right\}$$

(iv) **Elimination von mehrelementigen Nonterminalketten**

— Alle Produktionen der Form  $A \rightarrow B_1B_2 \dots B_n$  werden in die Produktionen  $A \rightarrow A_{n-1}B_n, A_{n-1} \rightarrow A_{n-2}B_{n-1}, \dots, A_2 \rightarrow B_1B_2$  zerteilt. Nach der Ersetzung sind alle längeren Nonterminalketten vollständig heruntergebrochen und die Chomsky-Normalform erreicht. —

$$P = \left\{ \begin{array}{l} S \rightarrow a \mid VA \mid AV \mid b \\ A \rightarrow VC \\ B \rightarrow AV \mid b \mid a \\ V \rightarrow a \\ C \rightarrow BB \end{array} \right\}$$

$$(c) P = \left\{ \begin{array}{l} S \rightarrow ABC \\ A \rightarrow aCD \\ B \rightarrow bCD \\ C \rightarrow D \mid \varepsilon \\ D \rightarrow C \end{array} \right\}$$

Der Automat auf flaci.com (FLACI: Formale Sprachen, abstrakte Automaten, Compiler und Interpreter) Ein Projekt der Hochschule Zittau/Görlitz und der Pädagogischen Hochschule Schwyz: flaci.com/Grxwcync2

Lösungsvorschlag

(i) **Elimination der  $\varepsilon$ -Regeln**

— Alle Regeln der Form  $A \rightarrow \varepsilon$  werden eliminiert. Die Ersetzung von  $A$  wird durch  $\varepsilon$  in allen anderen Regeln vorweggenommen.

$$P = \left\{ \begin{array}{l} S \rightarrow ABC \mid AB \\ A \rightarrow aCD \mid aD \\ B \rightarrow bCD \mid bD \\ C \rightarrow D \\ D \rightarrow C \mid \varepsilon \end{array} \right\}$$

In der letzten Regel entsteht ein neues  $\varepsilon$ . Es muss in der nächsten Iteration entfernt werden.

$$P = \left\{ \begin{array}{l} S \rightarrow ABC \mid AB \\ A \rightarrow aCD \mid aD \mid aC \mid a \\ B \rightarrow bCD \mid bD \mid bC \mid b \\ C \rightarrow D \\ D \rightarrow C \end{array} \right\}$$

(ii) **Elimination von Kettenregeln**

— Jede Produktion der Form  $A \rightarrow B$  mit  $A, B \in S$  wird als Kettenregel bezeichnet. Diese tragen nicht zur Produktion von Terminalzeichen bei und lassen sich ebenfalls eliminieren.

$$P = \left\{ \right.$$

$$S \rightarrow AB$$

$$A \rightarrow a$$

$$B \rightarrow b$$

}

$C$  und  $D$  sind nicht produktiv.  $C \rightarrow D$  und  $D \rightarrow C$  können gestrichen werden.

(iii) **Separation von Terminalzeichen**

— Jedes Terminalzeichen  $\sigma$ , das in Kombination mit anderen Symbolen auftaucht, wird durch ein neues Nonterminal  $S_\sigma$  ersetzt und die Menge der Produktionen durch die Regel  $S_\sigma \rightarrow \sigma$  ergänzt.

∅ Nichts zu tun

(iv) **Elimination von mehrelementigen Nonterminalketten**

— Alle Produktionen der Form  $A \rightarrow B_1 B_2 \dots B_n$  werden in die Produktionen  $A \rightarrow A_{n-1} B_n, A_{n-1} \rightarrow A_{n-2} B_{n-1}, \dots, A_2 \rightarrow B_1 B_2$  zerteilt. Nach der Ersetzung sind alle längeren Nonterminalketten vollständig heruntergebrochen und die Chomsky-Normalform erreicht.

∅ Nichts zu tun

## 66115 / 2021 / Frühjahr / Thema 2 / Aufgabe 1

### Mindestens eine Eins

Geben Sie für die folgenden Sprachen eine kontextfreie Grammatik an:

Die Wörter bestehen aus beliebig vielen Nullen, einer 1 sowie weiteren beliebig vielen Nullen oder Einsen.

Lösungsvorschlag

$$P = \{$$

$$S \rightarrow A1A$$

$$A \rightarrow 0A \mid 1A \mid \varepsilon$$

}

Der Automat auf [flaci.com](http://flaci.com) (FLACI: Formale Sprachen, abstrakte Automaten, Compiler und Interpreter) Ein Projekt der Hochschule Zittau/Görlitz und der Pädagogischen Hochschule Schwyz: [flaci.com/G539t1rgc](http://flaci.com/G539t1rgc)

## 66115 / 2021 / Frühjahr / Thema 2 / Aufgabe 1

### Kontextfreie Grammatik

- (a) Erstellen Sie eine Ableitung für die Wörter der Sprache zur vorgegeben Grammatik

$$G = (\{S, A, B\}, \{0, 1\}, P, S)$$

$$P = \left\{ \begin{array}{l} S \rightarrow A1B \\ A \rightarrow 0A \mid \varepsilon \\ B \rightarrow 0B \mid 1B \mid \varepsilon \end{array} \right\}$$

Der Automat auf flaci.com (FLACI: Formale Sprachen, abstrakte Automaten, Compiler und Interpreter) Ein Projekt der Hochschule Zittau/Görlitz und der Pädagogischen Hochschule Schwyz: flaci.com/Gilrgpemg

- 00101

Lösungsvorschlag

$$S \vdash A1B \vdash 0A1B \vdash 00A1B \vdash 001B \vdash 0010B \vdash 00101B \vdash 00101$$

- 1001

Lösungsvorschlag

$$S \vdash A1B \vdash 1B \vdash 10B \vdash 100B \vdash 1001B \vdash 1001$$

- (b) Erstellen Sie eine kontextfreie Grammatik, die alle Wörter mit gleich vielen 1's, gefolgt von gleich vielen 0's enthält.

Lösungsvorschlag

$$P = \left\{ \begin{array}{l} S \rightarrow 1S0 \mid \varepsilon \end{array} \right\}$$

Der Automat auf flaci.com (FLACI: Formale Sprachen, abstrakte Automaten, Compiler und Interpreter) Ein Projekt der Hochschule Zittau/Görlitz und der Pädagogischen Hochschule Schwyz: flaci.com/Grxmyw2ia

- (c) Erstellen Sie eine kontextfreie Grammatik, die alle regulären Ausdrücke über den Zeichen 0, 1 darstellt. (Beispiel:  $01^*(1+0)^0$  für einen möglichen regulären Ausdruck (Das +-Zeichen ist hier anstelle des Oder-Zeichens ( $\mid$ )))

Lösungsvorschlag

$$G = (\{S\}, \{1; 0; (;); +; *\}, P, S)$$

$$P = \left\{ \begin{array}{l} S \rightarrow \varepsilon \mid 0 \mid 1 \mid S * \mid (S) \mid SS \mid S + S \end{array} \right\}$$



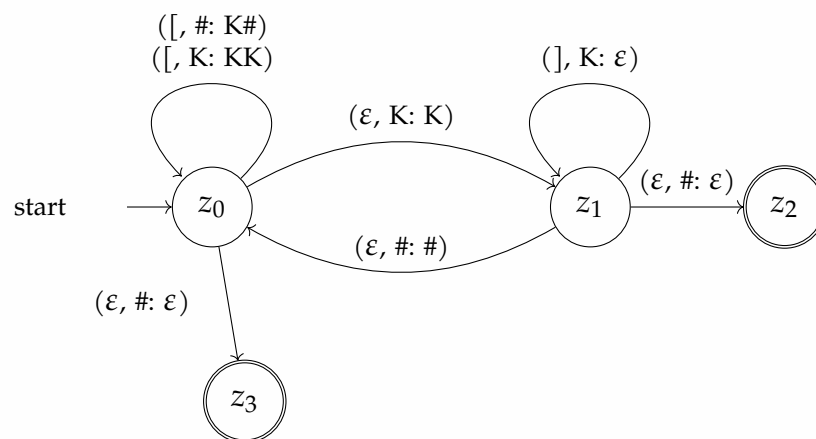
## 66115 / 2021 / Frühjahr / Thema 2 / Aufgabe 1

### Balancierte Klammern

Erstellen Sie einen Kellerautomaten der nur balancierte Klammerausdrücke mit eckigen Klammern akzeptiert.<sup>2</sup>

Lösungsvorschlag

$$K = (\{z_0, z_1, z_2, z_3\}, \{[, ]\}, \{\#, K\}, \delta, z_0, \#, \{z_2, z_3\})$$



Der Automat auf flaci.com (FLACI: Formale Sprachen, abstrakte Automaten, Compiler und Interpreter) Ein Projekt der Hochschule Zittau/Görlitz und der Pädagogischen Hochschule Schwyz: flaci.com/Apwobf482

## 66115 / 2021 / Frühjahr / Thema 2 / Aufgabe 1

### Kellerautomat

Erstellen Sie einen Kellerautomaten, der folgende Sprache

$$L = \{a^n b^n \mid n \in \mathbb{N}\}$$

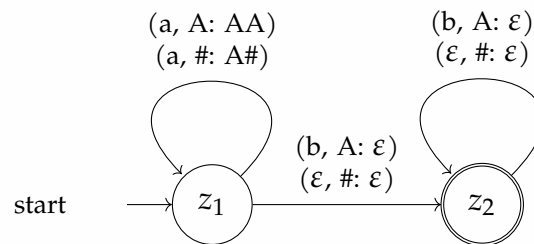
mit folgender Grammatik

$$G = (\{S\}, \{a, b\}, \{S \rightarrow aSb \mid ab\}, S)$$

erkennt.

<sup>2</sup><https://eecs.wsu.edu/~ananth/CptS317/Lectures/PDA.pdf> (Seite 9)

$$K = (\{z_1, z_2\}, \{a, b\}, \{\#, A\}, \delta, z_1, \#, \{z_2\})$$



Der Automat auf flaci.com (FLACI: Formale Sprachen, abstrakte Automaten, Compiler und Interpreter) Ein Projekt der Hochschule Zittau/Görlitz und der Pädagogischen Hochschule Schwyz: flaci.com/Ah5v17t52

Aktueller Zustand	Eingabe	Keller	Folgezustand	Keller
$z_1$	a	#	$z_1$	A#
$z_1$	a	A	$z_1$	AA
$z_1$	b	A	$z_2$	$\varepsilon$
$z_2$	b	A	$z_2$	$\varepsilon$
$z_2$	$\varepsilon$	#	$z_2$	#

## 66115 / 2021 / Frühjahr / Thema 2 / Aufgabe 1

### Grammatik in Kellerautomat umwandeln

Geben Sie für die folgenden Grammatiken  $G_i$  jeweils einen Kellerautomaten  $P_i$  an, der dieselbe Sprache besitzt wie die Grammatik:  $L(G_i) = L(P_i)$

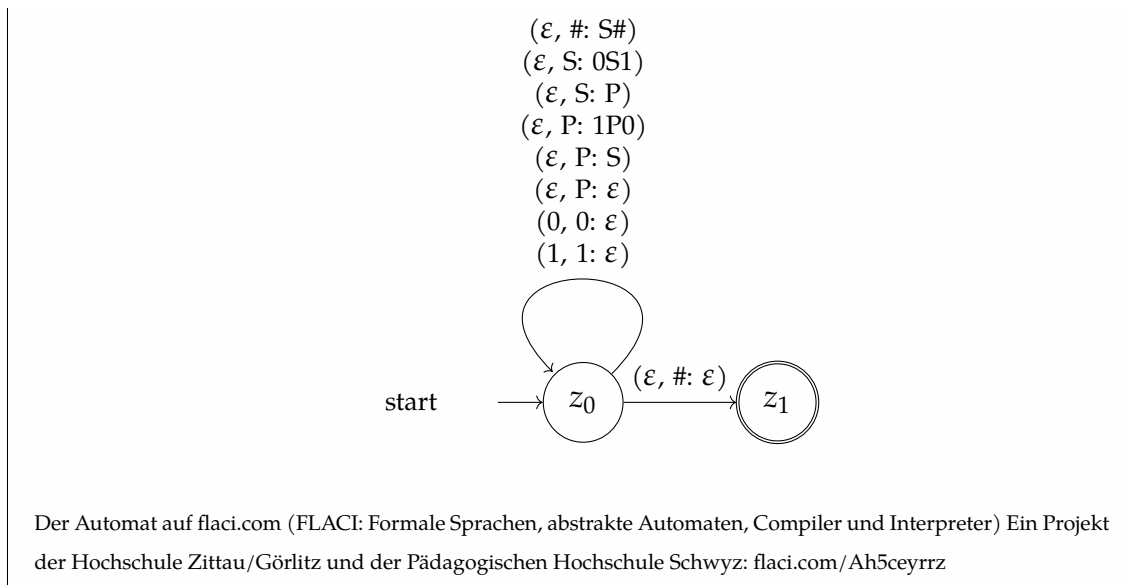
(a)  $P_1 = \left\{ \right.$

$$S \rightarrow 0S1 \mid P$$

$$P \rightarrow 1P0 \mid S \mid \varepsilon$$

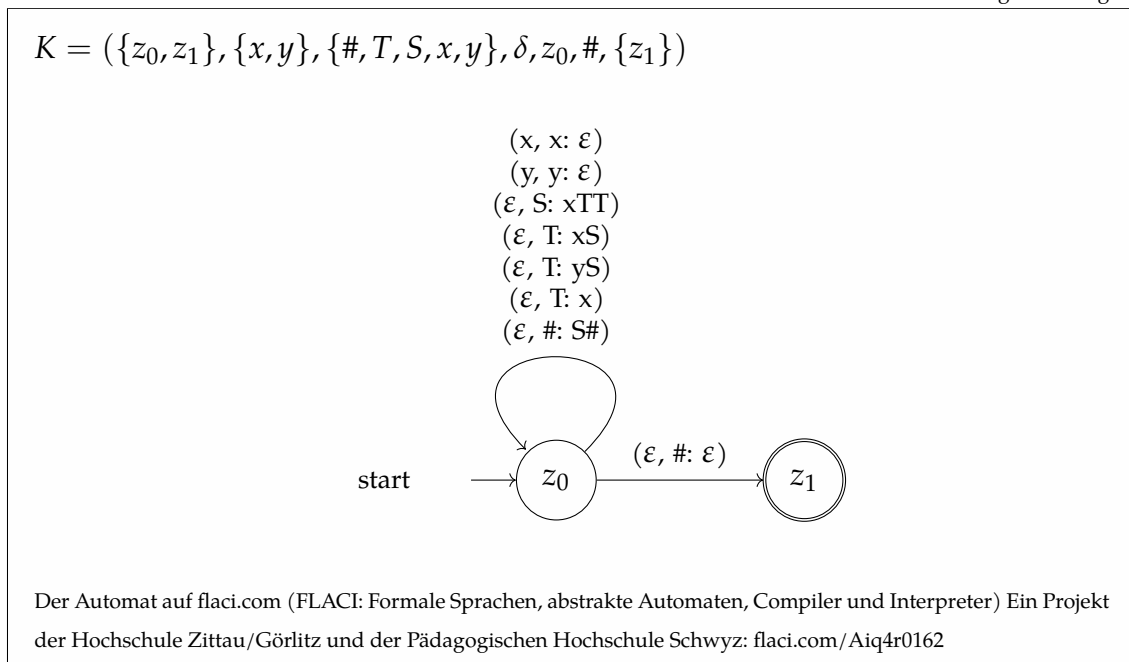
$\left. \right\}$

$$K = (\{z_0, z_1\}, \{0, 1\}, \{\#, S, P, 0, 1\}, \delta, z_0, \#, \{z_1\})$$



(b)  $P_2 = \left\{ \begin{array}{l} S \rightarrow xTT \\ T \rightarrow xS \mid yS \mid x \end{array} \right\}$

Lösungsvorschlag



(c)  $P_3 = \left\{ \right.$

$$S \rightarrow aB \mid bA \mid ABc \mid B$$

$$A \rightarrow SSa$$

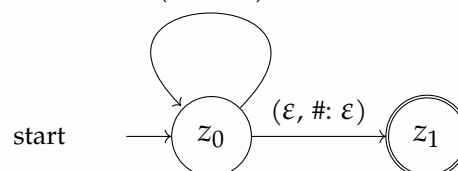
$$B \rightarrow cS \mid bB \mid b$$

}

Lösungsvorschlag

$$K = (\{z_0, z_1\}, \{a, b\}, \{\#, A, B, a, b\}, \delta, z_0, \#, \{z_1\})$$

$(a, a: \varepsilon)$   
 $(b, b: \varepsilon)$   
 $(c, c: \varepsilon)$   
 $(\varepsilon, S: aB)$   
 $(\varepsilon, S: bA)$   
 $(\varepsilon, S: ABc)$   
 $(\varepsilon, S: B)$   
 $(\varepsilon, A: SSa)$   
 $(\varepsilon, B: cS)$   
 $(\varepsilon, B: bB)$   
 $(\varepsilon, B: b)$   
 $(\varepsilon, \#: S\#)$

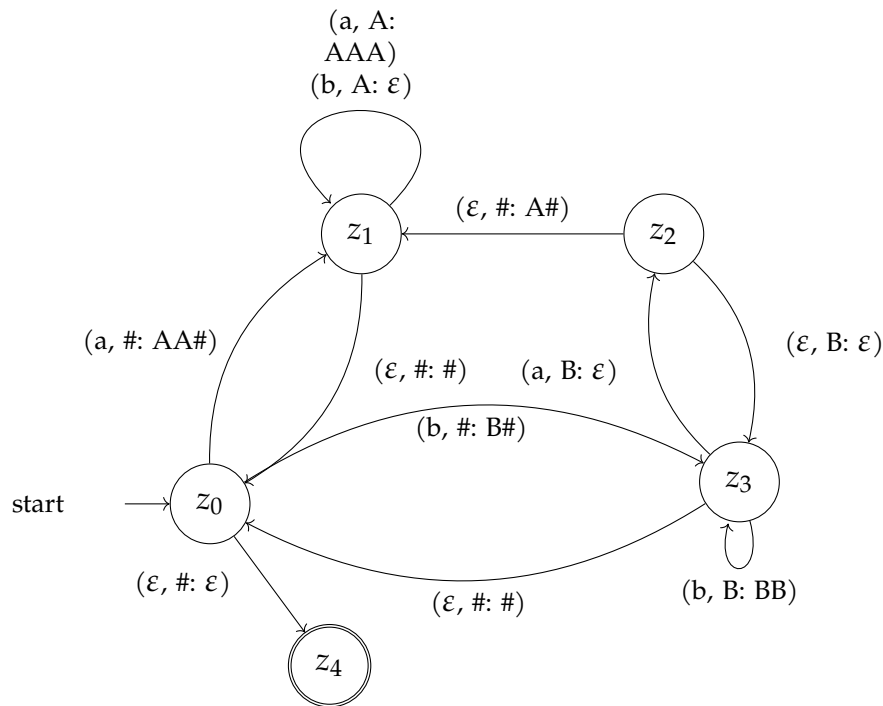


Der Automat auf flaci.com (FLACI: Formale Sprachen, abstrakte Automaten, Compiler und Interpreter) Ein Projekt der Hochschule Zittau/Görlitz und der Pädagogischen Hochschule Schwyz: [flaci.com/Ajh5y0s5r](http://flaci.com/Ajh5y0s5r)

## 66115 / 2021 / Frühjahr / Thema 2 / Aufgabe 1

Gegeben ist der folgende nichtdeterministische Kellerautomat mit

$$K = (\{z_0, z_1, z_2, z_3, z_4\}, \{a, b\}, \{\#, A, B\}, \delta, z_0, \#, \{z_4\})$$



Der Automat auf flaci.com (FLACI: Formale Sprachen, abstrakte Automaten, Compiler und Interpreter) Ein Projekt der Hochschule Zittau/Görlitz und der Pädagogischen Hochschule Schwyz: [flaci.com/Apk0ic3s9](http://flaci.com/Apk0ic3s9)

(a) Geben Sie für die folgenden Wörter, die in der Sprache enthalten sind, eine Berechnung (Folge von Konfigurationen) des Kellerautomaten an:

(i)  $w_1 = bab$

Lösungsvorschlag

$(z_0, bab, \#) \vdash (z_3, ab, B\#) \vdash (z_2, b, \#) \vdash (z_1, b, A\#) \vdash (z_1, \epsilon, \#) \vdash (z_0, \epsilon, \#) \vdash (z_4, \epsilon, \epsilon)$

(ii)  $w_2 = abb$

Lösungsvorschlag

$(z_0, abb, \#) \vdash (z_1, bb, AA\#) \vdash (z_1, b, A\#) \vdash (z_1, \epsilon, \#) \vdash (z_0, \epsilon, \#) \vdash (z_4, \epsilon, \epsilon)$

(iii)  $w_3 = abababbbb$

Lösungsvorschlag

$(z_0, abababbbb, \#) \vdash (z_1, bababbbb, AA\#) \vdash (z_1, ababbbb, A\#) \vdash (z_1, babb, AA\#) \vdash (z_1, abbbb, AA\#) \vdash (z_1, bbbb, AAA\#) \vdash (z_1, bbb, AAA\#) \vdash (z_1, bb, AA\#) \vdash (z_1, b, A\#) \vdash (z_1, \epsilon, \#) \vdash (z_0, \epsilon, \#) \vdash (z_4, \epsilon, \epsilon)$

(b) Charakterisiere die Wörter der Sprache in eigenen Worten.

$$L = \{ w \mid w \text{ enthält genau doppelt so viele } b\text{'s wie } a\text{'s} \}$$

## 66115 / 2021 / Frühjahr / Thema 2 / Aufgabe 1

### PDA

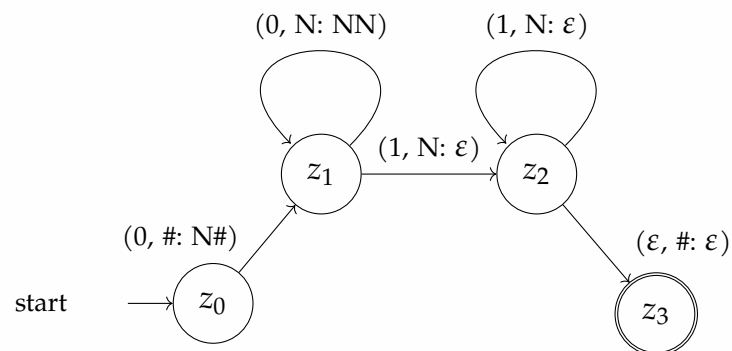
Erstellen Sie jeweils einen PDA, der die angegebenen Sprachen akzeptiert.

(a)  $L = \{ 0^n 1^n \mid n \in \mathbb{N} \}$

Lösungsvorschlag

$$K = (\{z_0, z_1, z_2, z_3\}, \{0, 1\}, \{\#, N\}, \delta, z_0, \#, \{z_3\})$$

$N = \text{Null}$

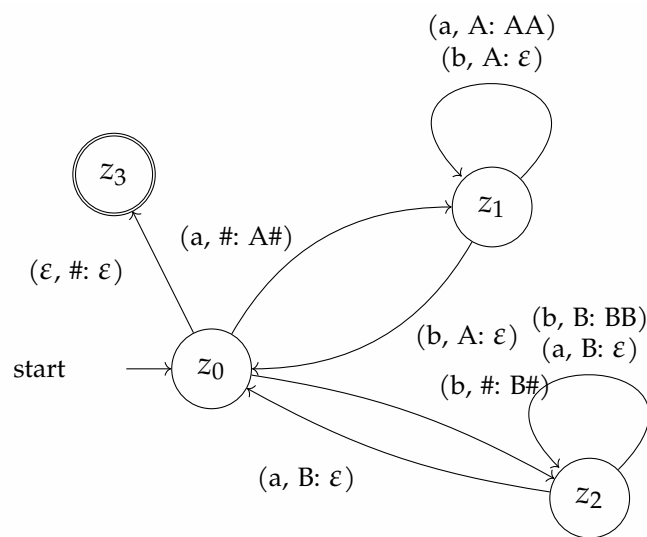


Der Automat auf flaci.com (FLACI: Formale Sprachen, abstrakte Automaten, Compiler und Interpreter) Ein Projekt der Hochschule Zittau/Görlitz und der Pädagogischen Hochschule Schwyz: [flaci.com/Aji1r8mf7](http://flaci.com/Aji1r8mf7)

(b) Alle Wörter, die gleich viele  $a$  wie  $b$  enthalten.

Lösungsvorschlag

$$K = (\{z_0, z_1, z_2, z_3\}, \{a, b\}, \{\#, A, B\}, \delta, z_0, \#, \{z_3\})$$



Der Automat auf [flaci.com](http://flaci.com) (FLACI: Formale Sprachen, abstrakte Automaten, Compiler und Interpreter) Ein Projekt der Hochschule Zittau/Görlitz und der Pädagogischen Hochschule Schwyz: [flaci.com/Arar6fos7](http://flaci.com/Arar6fos7)

(c) Alle Wörter, bei denen kein Präfix mehr Einsen wie Nullen hat.

Lösungsvorschlag

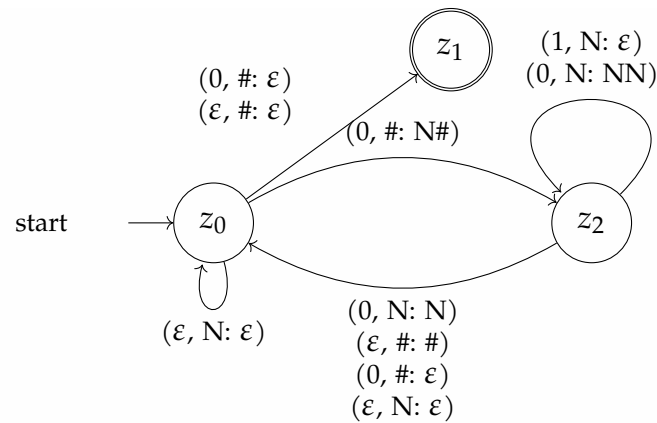
$$K = (\{z_0, z_1, z_2\}, \{0, 1\}, \{\#, N\}, \delta, z_0, \#, \{z_1\})$$

akzeptiert:

- 000
- 00101
- 01

nicht akzeptiert:

- 011
- 111
- 1



Der Automat auf flaci.com (FLACI: Formale Sprachen, abstrakte Automaten, Compiler und Interpreter) Ein Projekt der Hochschule Zittau/Görlitz und der Pädagogischen Hochschule Schwyz: [flaci.com/Af7rfyqqg](http://flaci.com/Af7rfyqqg)

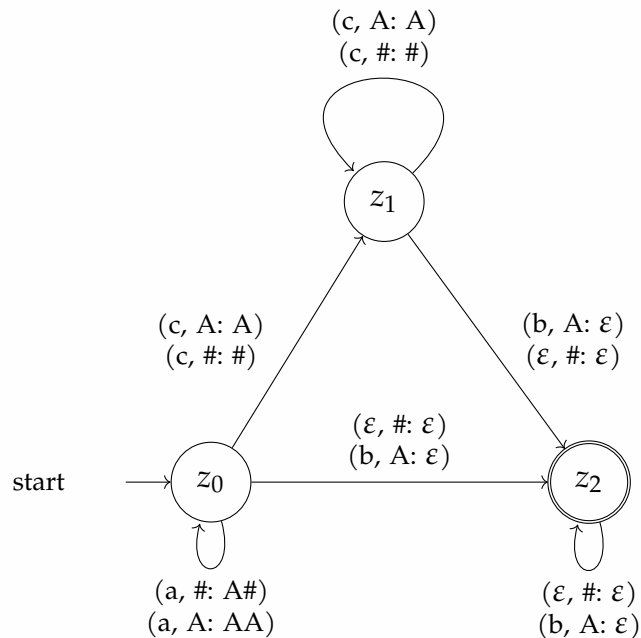
## 66115 / 2021 / Frühjahr / Thema 2 / Aufgabe 1

(a) Geben Sie einen Kellerautomaten an, der die folgende Sprache erkennt:

$$L = \{ a^n c^i b^n \mid n, i \in \mathbb{N}_0 \}$$

Lösungsvorschlag

$$K = (\{z_0, z_1, z_2\}, \{a, b, c\}, \{\#, A\}, \delta, z_0, \#, \{z_2\})$$



Der Automat auf flaci.com (FLACI: Formale Sprachen, abstrakte Automaten, Compiler und Interpreter) Ein Projekt der Hochschule Zittau/Görlitz und der Pädagogischen Hochschule Schwyz: [flaci.com/Apky9znog](http://flaci.com/Apky9znog)

Tabellenform:



Aktueller Zustand	Eingabe	Keller	Folgezustand	Keller
$z_0$	a	#	$z_0$	A#
$z_0$	a	A	$z_0$	AA
$z_0$	c	#	$z_1$	#
$z_0$	c	A	$z_1$	A
$z_0$	$\varepsilon$	#	$z_2$	$\varepsilon$
$z_0$	b	A	$z_2$	$\varepsilon$
$z_1$	c	#	$z_1$	#
$z_1$	c	A	$z_1$	A
$z_1$	$\varepsilon$	#	$z_2$	$\varepsilon$
$z_1$	b	A	$z_2$	$\varepsilon$
$z_2$	$\varepsilon$	#	$z_2$	$\varepsilon$
$z_2$	b	A	$z_2$	$\varepsilon$

(b) Geben Sie eine Grammatik für diese Sprache an.

Lösungsvorschlag

$P = \left\{ \begin{array}{l} S \rightarrow aSb \mid \varepsilon \mid c \mid cC \\ C \rightarrow cC \mid \varepsilon \end{array} \right\}$ <p>alternativ:</p> $P = \left\{ \begin{array}{l} S \rightarrow aSb \mid \varepsilon \mid C \\ C \rightarrow cC \mid \varepsilon \end{array} \right\}$
--

(c) Geben Sie Konfigurationsfolgen für die Erzeugung des Wortes an

- aacbb

Lösungsvorschlag

$$(z_0, aacbb, \#) \vdash (z_0, acbb, A\#) \vdash (z_0, cbb, AA\#) \vdash (z_1, bb, AA\#) \vdash (z_2, b, A\#) \vdash (z_2, \varepsilon, \#) \vdash (z_2, \varepsilon, \varepsilon)$$

- accb

Lösungsvorschlag

$$(z_0, accb, \#) \vdash (z_0, ccb, A\#) \vdash (z_1, cb, A\#) \vdash (z_2, b, A\#) \vdash (z_2, \varepsilon, \#) \vdash (z_2, \varepsilon, \varepsilon)$$

## 66115 / 2021 / Frühjahr / Thema 2 / Aufgabe 1

Erstellen Sie einen Kellerautomaten zu der Grammatik  $G = (\{S\}, \{0, 1\}, P, S)$  mit den folgenden Produktionsregeln

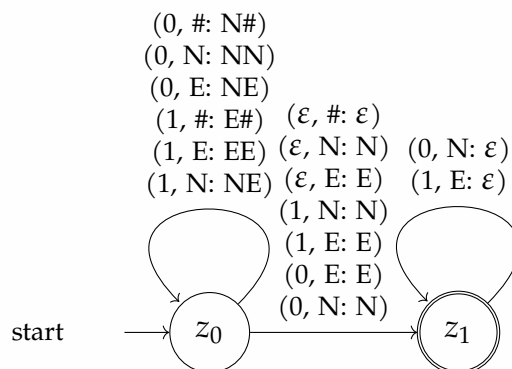
$$(a) P = \left\{ \begin{array}{l} S \rightarrow \varepsilon \mid 0 \mid 1 \mid 0S0 \mid 1S1 \end{array} \right\}$$

Lösungsvorschlag

 $K = (\{z_0, z_1\}, \{0, 1\}, \{\#, N, E\}, \delta, z_0, \#, \{z_1\})$ 

N = Null

E = Eins



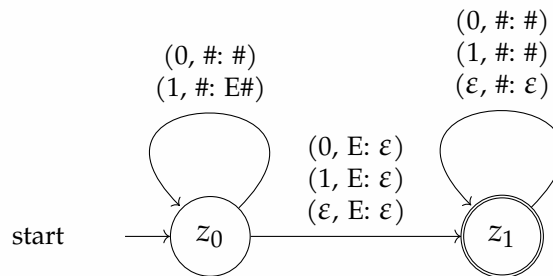
Der Automat auf flaci.com (FLACI: Formale Sprachen, abstrakte Automaten, Compiler und Interpreter) Ein Projekt der Hochschule Zittau/Görlitz und der Pädagogischen Hochschule Schwyz: flaci.com/Ahij8jnn7

$$(b) P = \left\{ \begin{array}{l} S \rightarrow A1B \\ A \rightarrow 0A \mid \varepsilon \\ B \rightarrow 0B \mid 1B \mid \varepsilon \end{array} \right.$$

Lösungsvorschlag

 $K = (\{z_0, z_1\}, \{0, 1\}, \{\#, E\}, \delta, z_0, \#, \{z_1\})$ 

E = Eins ist gesetzt



Der Automat auf flaci.com (FLACI: Formale Sprachen, abstrakte Automaten, Compiler und Interpreter) Ein Projekt der Hochschule Zittau/Görlitz und der Pädagogischen Hochschule Schwyz: flaci.com/Ar3imp8a7

## 66115 / 2021 / Frühjahr / Thema 2 / Aufgabe 1

### Pumping-Lemma

Gegeben sei die Sprachen

$$L = \{a^n b^n c^n \mid n \in \mathbb{N}\}$$

Weisen Sie nach, dass  $L$  nicht kontextfrei ist.

#### Exkurs: Pumping-Lemma für Kontextfreie Sprachen

Es sei  $L$  eine kontextfreie Sprache. Dann gibt es eine Zahl  $j$ , sodass sich alle Wörter  $\omega \in L$  mit  $|\omega| \geq j$  zerlegen lassen in  $\omega = uvwxy$ , sodass die folgenden Eigenschaften erfüllt sind:

- (a)  $|vx| \geq 1$  (Die Wörter  $v$  und  $x$  sind nicht leer.)
- (b)  $|vwx| \leq j$  (Die Wörter  $v$ ,  $w$  und  $x$  haben zusammen höchstens die Länge  $j$ .)
- (c) Für alle  $i \in \mathbb{N}_0$  gilt  $uv^iwx^iy \in L$  (Für jede natürliche Zahl (mit 0)  $i$  ist das Wort  $uv^iwx^iy$  in der Sprache  $L$ )

Lösungsvorschlag

Also gibt es eine Pumpzahl. Sie sei  $j$ . (Wähle geschickt ein „langes“ Wort...)  $a^j b^j c^j$  ist ein Wort aus  $L$ , das sicher länger als  $j$  ist.

Da  $L$  kontextfrei ist, muss es nach dem Pumping-Lemma auch für dieses Wort eine beliebige Zerlegung geben:

$$a^j b^j c^j = uvwxy \text{ mit } |vx| \geq 1 \text{ und } |vwx| \leq j$$

Weil  $vwx$  höchstens  $j$  lang ist, kann es nie  $a$ 's und  $c$ 's zugleich enthalten (es stehen  $j$   $b$ 's dazwischen!).

Andererseits enthält  $vx$  mindestens ein Zeichen. Das Wort  $\omega = uv^0wx^0y = uwy$

enthält dann nicht mehr gleich viele  $a$ 's,  $b$ 's und  $c$ 's. (Widerspruch)!

Die Behauptung ist falsch.

$\Rightarrow L$  ist nicht kontextfrei!

Pumping-Lemma  
(Kontextfreie Sprache)

## 66115 / 2021 / Frühjahr / Thema 2 / Aufgabe 1

### Pumping-Lemma

Zeigen Sie jeweils, dass die angegebene Sprache nicht kontextfrei ist:

#### Exkurs: Pumping-Lemma für Kontextfreie Sprachen

Es sei  $L$  eine kontextfreie Sprache. Dann gibt es eine Zahl  $j$ , sodass sich alle Wörter  $\omega \in L$  mit  $|\omega| \geq j$  zerlegen lassen in  $\omega = uvwxy$ , sodass die folgenden Eigenschaften erfüllt sind:

- (a)  $|vx| \geq 1$  (Die Wörter  $v$  und  $x$  sind nicht leer.)
- (b)  $|vwx| \leq j$  (Die Wörter  $v$ ,  $w$  und  $x$  haben zusammen höchstens die Länge  $j$ .)
- (c) Für alle  $i \in \mathbb{N}_0$  gilt  $uv^iwx^iy \in L$  (Für jede natürliche Zahl (mit 0)  $i$  ist das Wort  $uv^iwx^iy$  in der Sprache  $L$ )

(a)  $L_1 = \{ ww \mid w \in \{a, b\}^* \}$

Lösungsvorschlag

Sei  $L_1$  kontextfrei. Dann existiert nach dem Pumping-Lemma eine Zahl  $j$ , so dass für jedes Wort  $\omega \in L_1$  mit  $|\omega| \geq j$  eine Zerlegung  $\omega = uvwxy$  existiert, für die gilt:  $|vx| > 0$ ,  $|vwx| \leq n$  und für jedes  $i \in \mathbb{N}$  ist  $uv^iwx^iy \in L_1$ .

Wähle  $\omega = a^n b^n a^n b^n$ . Dann gibt es für jede Zerlegung  $\omega = uvxyz$  mit den obigen Bedingungen zwei Möglichkeiten:

- $vwx$  besteht aus  $a^j b^k$  mit  $j + k > 0$ .
- $vwx$  besteht aus  $b^j a^k$  mit  $j + k > 0$ .

Dann ist in beiden Fällen  $uv^0xy^0z \notin L_1$ .

(b)  $L_2 = \{ a^k b^l c^m \mid k > l > m; k, l, m \in \mathbb{N} \}$

Lösungsvorschlag

Sei  $L_2$  kontextfrei. Dann existiert nach dem Pumping-Lemma eine Zahl  $j$ , so dass für jedes Wort  $\omega \in L_2$  mit  $|\omega| \geq j$  eine Zerlegung  $\omega = uvwxy$  existiert, für die gilt:  $|vx| > 0$ ,  $|vwx| \leq j$  und für jedes  $i \in \mathbb{N}$  ist  $uv^iwx^iy \in L_2$ .

Wähle  $\omega = a^n b^{n-1} c^{n-2}$ . Dann gibt es für jede Zerlegung  $\omega = uvwxy$  mit den obigen Bedingungen zwei Möglichkeiten:

- $vwx$  enthält kein  $a$ . Dann ist  $uv^2wx^2y \notin L_2$ .

-  $vwx$  enthält mindestens ein  $a$ . Dann ist  $uv^0wx^0y \notin L_2$ .

Pumping-Lemma  
(Kontextfreie Sprache)

## 66115 / 2021 / Frühjahr / Thema 2 / Aufgabe 1

### Pumping-Lemma

Zeigen Sie, dass die folgenden Sprachen nicht kontextfrei sind:

#### Exkurs: Pumping-Lemma für Kontextfreie Sprachen

Es sei  $L$  eine kontextfreie Sprache. Dann gibt es eine Zahl  $j$ , sodass sich alle Wörter  $\omega \in L$  mit  $|\omega| \geq j$  zerlegen lassen in  $\omega = uvwxy$ , sodass die folgenden Eigenschaften erfüllt sind:

- (a)  $|vx| \geq 1$  (Die Wörter  $v$  und  $x$  sind nicht leer.)
- (b)  $|vwx| \leq j$  (Die Wörter  $v, w$  und  $x$  haben zusammen höchstens die Länge  $j$ .)
- (c) Für alle  $i \in \mathbb{N}_0$  gilt  $uv^iwx^iy \in L$  (Für jede natürliche Zahl (mit 0)  $i$  ist das Wort  $uv^iwx^iy$  in der Sprache  $L$ )

-  $L = \{ a^n b^n c^{2n} \mid n \in \mathbb{N} \}$

Lösungsvorschlag

Annahme:  $L$  ist kontextfrei.

$\forall \omega \in L: \omega = uvwxy$

$j \in \mathbb{N}: |\omega| \geq j$

$\omega = a^j b^j c^{2j}: |\omega| = 4j > j$

Damit gilt:  $|vwx| \leq j, |vx| \geq 1$

Zu zeigen: Keine Möglichkeit der Zerlegung, damit  $\omega' \in L$

**1. Fall**  $vwx$  enthält nur  $a$ 's

o. E. d. A. (ohne Einschränkung der Allgemeinheit) stecken alle  $a$ 's in der Zerlegung  $vwx$ ,  $u$  ist leer

$u : \varepsilon$

$v : a^l$

$w : a^{j-(l+m)}$

$x : a^m$

$y : b \dots bc \dots c$

$v^2wx^2y$

$a^{2l}a^{j-(l+m)}a^{2m}b^jc^{2j} =$

Nebenrechnung:  $2l + j - (l + m) + 2m = j + l + m > j$ , da  $|vx| \geq 1 \rightarrow l + m \geq 1$

$\Rightarrow \omega' = uv^2wx^2y \notin L$

**2. Fall**  $vwx$  enthalten  $a$ 's und  $b$ 's

o. E. d. A.  $|v|_a = |x|_b$

$u: a^p v: a^l w: a^{j-(p+l)} b^{j-(l+r)} x: b^l y: b^r c^{2j}$

$\Rightarrow uv^0wx^0v$

Nebenrechnung:

$a's: p + j - (l + p) = j - l$

$b's: j - (l + r) = j - l$

ist falsch, da  $j - l$  echt kleiner ist, da  $|vx| \geq 1 \rightarrow l \geq 1$

$\Rightarrow \omega' \notin L$

**3. Fall**  $vw x$  enthält nur  $b's$

analog zu Fall 1

**4. Fall**  $vw x$  enthält nur  $b's$  und  $c's$

analog zu Fall 2

**5. Fall**  $vw x$  enthält nur  $c's$

analog zu Fall 1

$\Rightarrow$  Es gibt keine Zerlegung, sodass  $\forall i \in \mathbb{N}_0$

$\Rightarrow$  Annahme ist falsch

$\Rightarrow L$  ist nicht kontextfrei

$$- L = \{ a^n b^{n^2} \mid n \in \mathbb{N} \}$$

Lösungsvorschlag

Annahme:  $L$  kontextfrei

$\Rightarrow$  Pumping-Lemma:  $j \in \mathbb{N}: |w| \geq j$

$$\omega = a^j b^{j^2}$$

$$j + j^2 > j$$

**1. Fall**  $vw x$  enthält nur  $a's$

$\Rightarrow$  ungleich viele  $a's$  wie  $b's$  als Quadrat

$\Rightarrow \omega' \notin E$

**2. Fall**  $vw x$  enthält nur  $b's$

$\Rightarrow$  analog zu Fall 1

$\Rightarrow \omega' \notin E$

**3. Fall**  $vw x$  enthält  $a's$  und  $b's$

o. E. d. A.  $v$  nur  $a's$ ;  $x$  nur  $b's$

$u: a^{j-(l+m)}$

$v: a^l$

$w: a^m b^n$

$x: b^{l^2}$

$$y: b^{j^2 - (n + l^2)}$$

$$\Rightarrow uv^0wx^0y = \omega'$$

$$a: j - (l + m) + 0 \cdot l + m = j - l$$

$$b: n - 0 \cdot l^2 + j^2 - (n + l^2) = j^2 - l^2 = (j - l)(j + l) \neq (j - l)(j - l)$$

$$\Rightarrow \in L$$

## 46115 / 2019 / Herbst / Thema 1 / Aufgabe 2

(a) Betrachten Sie die folgenden Sprachen:

$$L_1 = \{ a^n b^m c^k \mid n, m \in \mathbb{N} \}$$

$L_2 = \{ a^n b^m c^k \mid n = m \}$  Zeigen Sie für  $L_1$  und  $L_2$ , ob sie kontextfrei sind oder nicht. Für den Beweis von Kontext-Freiheit in dieser Frage reicht die Angabe eines Automaten oder einer Grammatik. (Beschreiben Sie dann die Konstruktionsidee des Automaten oder der Grammatik.) Für den Beweis von Nicht-Kontext-Freiheit verwenden Sie eine der üblichen Methoden.

(b) Eine kontextfreie Grammatik ist in Chomsky-Normalform, falls die folgenden Bedingungen erfüllt sind:

- alle Regeln sind von der Form  $X \rightarrow YZ$  oder  $X \rightarrow a$  mit Nichtterminalzeichen  $X, Y, Z$  und Terminalzeichen  $a$ ,
- alle Nichtterminalzeichen sind erreichbar vom Startsymbol und
- alle Nichtterminalzeichen sind erzeugend, d. h. für jedes Nichtterminalzeichen  $X$  gibt es ein Wort  $w$  über dem Terminalalphabet, so dass  $X \Rightarrow^* w$ .

Bringen Sie die folgende Grammatik in Chomsky-Normalform.

$$P = \left\{ \begin{array}{l} S \rightarrow AAB \mid CD \mid abc \\ A \rightarrow AAAA \mid a \\ B \rightarrow BB \mid S \\ C \rightarrow CCC \mid CC \\ D \rightarrow d \end{array} \right\}$$

Der Automat auf flaci.com (FLACI: Formale Sprachen, abstrakte Automaten, Compiler und Interpreter) Ein Projekt der Hochschule Zittau/Görlitz und der Pädagogischen Hochschule Schwyz: [flaci.com/Gf7f9tp7z](http://flaci.com/Gf7f9tp7z)

Das Startsymbol der Grammatik ist  $S$ , das Terminalalphabet ist  $a, b, c, d$  und die Menge der Nichtterminalzeichen ist  $S, A, B, C, D$ .

**(i) Elimination der  $\varepsilon$ -Regeln**

— Alle Regeln der Form  $A \rightarrow \varepsilon$  werden eliminiert. Die Ersetzung von  $A$  wird durch  $\varepsilon$  in allen anderen Regeln vorweggenommen. —

$\emptyset$  Nichts zu tun

**(ii) Elimination von Kettenregeln**

— Jede Produktion der Form  $A \rightarrow B$  mit  $A, B \in S$  wird als Kettenregel bezeichnet. Diese tragen nicht zur Produktion von Terminalzeichen bei und lassen sich ebenfalls eliminieren. —

Eine rechte Seite in der  $C$  vorkommt, lässt sich wegen  $\{C \rightarrow CCC \mid CC\}$  nicht ableiten, weil es zu einer Endlosschleife kommt. Wir entfernen die entsprechenden Regeln.

$$P = \left\{ \begin{array}{l} S \rightarrow AAB \mid abc \\ A \rightarrow AAAA \mid a \\ B \rightarrow BB \mid AAB \mid abc \end{array} \right\}$$

**(iii) Separation von Terminalzeichen**

— Jedes Terminalzeichen  $\sigma$ , das in Kombination mit anderen Symbolen auftaucht, wird durch ein neues Nonterminal  $S_\sigma$  ersetzt und die Menge der Produktionen durch die Regel  $S_\sigma \rightarrow \sigma$  ergänzt. —

$$P = \left\{ \begin{array}{l} S \rightarrow AAB \mid T_a T_b T_c \\ A \rightarrow AAAA \mid a \\ B \rightarrow BB \mid AAB \mid T_a T_b T_c \\ T_a \rightarrow a \\ T_b \rightarrow b \\ T_c \rightarrow c \end{array} \right\}$$

**(iv) Elimination von mehrelementigen Nonterminalketten**

— Alle Produktionen der Form  $A \rightarrow B_1 B_2 \dots B_n$  werden in die Produktionen  $A \rightarrow A_{n-1} B_n, A_{n-1} \rightarrow A_{n-2} B_{n-1}, \dots, A_2 \rightarrow B_1 B_2$  zerteilt. Nach der Ersetzung sind alle längeren Nonterminalketten vollständig heruntergebrochen und die Chomsky-Normalform erreicht. —

$$P = \left\{ \right.$$



$$\begin{aligned}
 S &\rightarrow AS_1 \mid T_a S_2 \\
 A &\rightarrow AA_1 \mid a \\
 B &\rightarrow BB \mid AS_1 \mid T_a S_2 \\
 T_a &\rightarrow a \\
 T_b &\rightarrow b \\
 T_c &\rightarrow c \\
 S_1 &\rightarrow AB \\
 S_2 &\rightarrow T_b T_c \\
 A_1 &\rightarrow AA_2 \\
 A_2 &\rightarrow AA
 \end{aligned}$$

}

**46115 / 2021 / Frühjahr / Thema 1 / Teilaufgabe 1 / Aufgabe 2**

- (a) Verwenden Sie den Algorithmus von Cocke, Younger und Kasami (CYK-Algorithmus), um für die folgende kontextfreie Grammatik  $G = (V, D, P, 5)$  mit Variablen  $V = \{S, A, B, C, D\}$ , Terminalzeichen  $\Sigma = \{a, b\}$ , Produktionen

$$P = \{S \rightarrow SB \mid AC \mid a, A \rightarrow a, B \rightarrow b, C \rightarrow DD \mid AB, D \rightarrow AB \mid DC \mid CD\}$$

und Startsymbol  $S$  zu prüfen, ob das Wort  $aabababb$  in der durch  $G$  erzeugten Sprache liegt. Erläutern Sie dabei Ihr Vorgehen und den Ablauf des CYK-Algorithmus.

- (b) Mit  $a^i$ , wobei  $i \in \mathbb{N}_0 = \{0, 1, 2, \dots\}$ , wird das Wort bezeichnet, das aus der  $i$ -fachen Wiederholung des Zeichens  $a$  besteht (d. h.  $a^0 = \epsilon$  und  $a^i = aa^{i-1}$ , wobei  $\epsilon$  das leere Wort ist).

Sei die Sprache  $L$  definiert über dem Alphabet  $\{a, b\}$  als

$$L = \{a^i b^m \mid i \in \mathbb{N}_0, m > 1\}.$$

Zeigen Sie, dass die Sprache  $L$  nicht vom Typ 3 der Chomsky-Hierarchie (d. h. nicht regulär) ist.

**66115 / 2012 / Frühjahr / Thema 1 / Aufgabe 3**

Beweisen Sie, dass folgende Sprache kontextfrei, aber nicht regulär ist.

$$C = \{a^n b^m \mid n \geq m \geq 1\}$$

Lösungsvorschlag

**Nachweis Kontextfrei über Grammatik**

$$G = (\{S\}, \{a, b\}, P, S)$$

$$P = \{$$

$$S \rightarrow aSb \mid aS \mid ab$$

}

- Regel 1:  $aSb$
- Regel 2:  $aS$
- Regel 3:  $ab$

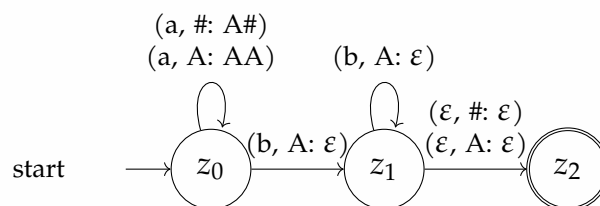
$$ab: S \xrightarrow{3} ab$$

$$a^n b: S \xrightarrow[n-1]{2} a^{n-1} S \xrightarrow{3} a^{n-1} ab$$

$$a^n b^m: S \xrightarrow[m-1]{1} a^{m-1} S b^{m-1} \xrightarrow[n-(m-1)]{2} a^{n-1} S b^{m-1} \xrightarrow{3} a^n b^m$$

$$\Rightarrow L(G) = C$$

### Nachweis Kontextfrei über Kellerautomat



Der Automat auf [flaci.com](http://flaci.com) (FLACI: Formale Sprachen, abstrakte Automaten, Compiler und Interpreter) Ein Projekt der Hochschule Zittau/Görlitz und der Pädagogischen Hochschule Schwyz: [flaci.com/Aji151myg](http://flaci.com/Aji151myg)

### Nachweis: C nicht regulär

C sei regulär

$\Rightarrow$  Pumping-Lemma für C erfüllt

$j$  sei die Pumping-Zahl ( $j \in \mathbb{N}$ )

$\omega \in C: \omega = a^j b^j$

$\omega = uvw$

Dann gilt:

- $|v| \geq 1$
- $|uv| \leq j$
- $uv^i w \in C$  für alle  $i \in \mathbb{N}_0$

In  $uv$  können nur  $a$ 's vorkommen  
 $\Rightarrow$  In  $v$  muss mindestens ein  $a$  vorkommen  
 $\Rightarrow uv^0w = a^l(a^{j-l})^0b^j ((a^{j-l})^0 = \varepsilon)$   
 $\Rightarrow$  In  $\omega'$  sind nur  $l$  viele  $a$ 's, Da  $l < j$ ,  $\omega' \notin C$ ,  
 $\Rightarrow$  Widerspruch zur Annahme  
 $\Rightarrow C$  nicht regulär

## 66115 / 2012 / Frühjahr / Thema 1 / Aufgabe 4

Gegeben ist die kontextfreie Grammatik  $G = (V, \Sigma, P, S)$  mit  $\Sigma = \{a, b\}$ ,  $N = \{S, A, B\}$  und

$P = \{$

$S \rightarrow A$   
 $S \rightarrow B$   
 $A \rightarrow aAb$   
 $B \rightarrow AA$   
 $B \rightarrow bBa$   
 $A \rightarrow a$

$\}$

Der Automat auf [flaci.com](http://flaci.com) (FLACI: Formale Sprachen, abstrakte Automaten, Compiler und Interpreter) Ein Projekt der Hochschule Zittau/Görlitz und der Pädagogischen Hochschule Schwyz: [flaci.com/Gr3rgt2vg](http://flaci.com/Gr3rgt2vg)

Geben Sie eine äquivalente Grammatik in Chomsky-Normalform an.

Lösungsvorschlag

Kann auch so geschrieben werden:

$P = \{$

$S \rightarrow A \mid B$   
 $A \rightarrow aAb \mid a$   
 $B \rightarrow AA \mid bBa$

$\}$

### (a) Elimination der $\varepsilon$ -Regeln

— Alle Regeln der Form  $A \rightarrow \varepsilon$  werden eliminiert. Die Ersetzung von  $A$  wird durch  $\varepsilon$  in allen anderen Regeln vorweggenommen. \_\_\_\_\_

$\emptyset$  Nichts zu tun

### (b) Elimination von Kettenregeln

— Jede Produktion der Form  $A \rightarrow B$  mit  $A, B \in S$  wird als Kettenregel bezeichnet. Diese tragen nicht zur Produktion von Terminalzeichen bei und lassen sich ebenfalls eliminieren. \_\_\_\_\_

$$P = \left\{ \begin{array}{l} S \rightarrow aAb \mid a \mid AA \mid bBa \\ A \rightarrow aAb \mid a \\ B \rightarrow AA \mid bBa \end{array} \right\}$$

**(c) Separation von Terminalzeichen**

— Jedes Terminalzeichen  $\sigma$ , das in Kombination mit anderen Symbolen auftaucht, wird durch ein neues Nonterminal  $S_\sigma$  ersetzt und die Menge der Produktionen durch die Regel  $S_\sigma \rightarrow \sigma$  ergänzt. —

$$P = \left\{ \begin{array}{l} S \rightarrow T_aAT_b \mid a \mid AA \mid T_bBT_a \\ A \rightarrow T_aAT_b \mid a \\ B \rightarrow AA \mid T_bBT_a \\ T_a \rightarrow a \\ T_b \rightarrow b \end{array} \right\}$$

**(d) Elimination von mehrelementigen Nonterminalketten**

— Alle Produktionen der Form  $A \rightarrow B_1B_2 \dots B_n$  werden in die Produktionen  $A \rightarrow A_{n-1}B_n, A_{n-1} \rightarrow A_{n-2}B_{n-1}, \dots, A_2 \rightarrow B_1B_2$  zerteilt. Nach der Ersetzung sind alle längeren Nonterminalketten vollständig heruntergebrochen und die Chomsky-Normalform erreicht. —

$$P = \left\{ \begin{array}{l} S \rightarrow T_aC \mid a \mid AA \mid T_bD \\ A \rightarrow T_aC \mid a \\ B \rightarrow AA \mid T_bD \\ T_a \rightarrow a \\ T_b \rightarrow b \\ C \rightarrow AT_b \\ D \rightarrow BT_a \end{array} \right\}$$

**66115 / 2013 / Frühjahr / Thema 1 / Aufgabe 2**

Gegeben sei die Grammatik  $G = (\{S, A, B, C\}, \{a, b\}, P, S)$  und

$$P = \left\{ \begin{array}{l} S \rightarrow AB \\ S \rightarrow CS \\ A \rightarrow BC \\ A \rightarrow BB \\ A \rightarrow a \\ B \rightarrow AC \\ B \rightarrow b \\ C \rightarrow AA \\ C \rightarrow BA \end{array} \right\}$$

Der Automat auf flaci.com (FLACI: Formale Sprachen, abstrakte Automaten, Compiler und Interpreter) Ein Projekt der Hochschule Zittau/Görlitz und der Pädagogischen Hochschule Schwyz: flaci.com/Gr46a6j0a

$L = L(G)$  ist die von  $G$  erzeugte Sprache.

(a) Zeigen Sie, dass  $G$  mehrdeutig ist.

Lösungsvorschlag

Das Wort  $baab$  kann in zwei verschiedenen Ableitungen hergeleitet werden:

(i)  $S \vdash AB \vdash BCB \vdash bCB \vdash bAAB \vdash baAB \vdash baaB \vdash baab$

(ii)  $S \vdash CS \vdash BAS \vdash bAS \vdash baS \vdash baAB \vdash baaB \vdash baab$

(b) Entscheiden Sie mithilfe des Algorithmus von Cocke, Younger und Kasami (CYK), ob das Wort  $w = babaaa$  zur Sprache  $L$  gehört. Begründen Sie Ihre Entscheidung.

Lösungsvorschlag

b	a	b	a	a	a
B	A	B	A	A	A
C	S	C	C	C	
-	B	A	B		
A	C	A,C			
A,C	B,C,A				
A,C,B					

$\Rightarrow babaaa \notin L(G)$

Das Startsymbol  $S$  ist nicht in der Zelle  $V(1,5) = \{A, C, B\}$  enthalten.

(c) Geben Sie eine Ableitung für  $w = babaaa$  an.

$$A \vdash BB \vdash bB \vdash bAC \vdash baC \vdash baAA \vdash baBCA \vdash babCA \vdash babAAA \vdash babaAA \vdash babaaA \vdash babaaa$$

## 66115 / 2016 / Frühjahr / Thema 1 / Aufgabe 2

Betrachten Sie die folgende Grammatik  $G = (\{S, A\}, \{0, 1, 2\}, P, S)$  mit

$$P = \left\{ \begin{array}{l} S \rightarrow 0S0 \mid 1S1 \mid 2A2 \mid 0 \mid 1 \mid \varepsilon \\ A \rightarrow A2 \end{array} \right\}$$

Der Automat auf [flaci.com](http://flaci.com) (FLACI: Formale Sprachen, abstrakte Automaten, Compiler und Interpreter) Ein Projekt der Hochschule Zittau/Görlitz und der Pädagogischen Hochschule Schwyz: [flaci.com/Gf6scqja9](http://flaci.com/Gf6scqja9)

Konstruieren Sie für die Grammatik  $G$  schrittweise eine äquivalente Grammatik in Chomsky-Normalform. Geben Sie für jeden einzelnen Schritt des Verfahrens das vollständige Zwischenergebnis an und erklären Sie kurz, was in dem Schritt getan wurde.

Lösungsvorschlag

Die Regeln  $\{S \rightarrow 2A2\}$  und  $\{A \rightarrow A2\}$  können gelöscht werden, da es keine Regel  $\{A \rightarrow \varepsilon\}$  oder  $\{A \rightarrow S\}$  gibt. So erhalten wir:

$$P = \left\{ \begin{array}{l} S \rightarrow 0S0 \mid 1S1 \mid 0 \mid 1 \mid \varepsilon \end{array} \right\}$$

### (a) Elimination der $\varepsilon$ -Regeln

— Alle Regeln der Form  $A \rightarrow \varepsilon$  werden eliminiert. Die Ersetzung von  $A$  wird durch  $\varepsilon$  in allen anderen Regeln vorweggenommen. —

falls  $S \rightarrow \varepsilon \in P$  neuen Startzustand  $S_1$  einführen

$$P = \left\{ \begin{array}{l} S \rightarrow 0S0 \mid 1S1 \mid 0 \mid 1 \mid 00 \mid 11 \\ S_1 \rightarrow \varepsilon \mid S \end{array} \right\}$$

### (b) Elimination von Kettenregeln

— Jede Produktion der Form  $A \rightarrow B$  mit  $A, B \in S$  wird als Kettenregel bezeichnet. Diese tragen nicht zur Produktion von Terminalzeichen bei und lassen sich ebenfalls eliminieren. —

∅ Nichts zu tun

**(c) Separation von Terminalzeichen**

— Jedes Terminalzeichen  $\sigma$ , das in Kombination mit anderen Symbolen auftaucht, wird durch ein neues Nonterminal  $S_\sigma$  ersetzt und die Menge der Produktionen durch die Regel  $S_\sigma \rightarrow \sigma$  ergänzt. —

N = Null E = Eins

$$P = \left\{ \begin{array}{l} S \rightarrow NSN \mid ESE \mid 0 \mid 1 \mid NN \mid EE \\ S_1 \rightarrow \varepsilon \mid S \\ A \rightarrow AZ \\ N \rightarrow 0 \\ E \rightarrow 1 \end{array} \right\}$$

**(d) Elimination von mehrelementigen Nonterminalketten**

— Alle Produktionen der Form  $A \rightarrow B_1 B_2 \dots B_n$  werden in die Produktionen  $A \rightarrow A_{n-1} B_n, A_{n-1} \rightarrow A_{n-2} B_{n-1}, \dots, A_2 \rightarrow B_1 B_2$  zerteilt. Nach der Ersetzung sind alle längeren Nonterminalketten vollständig heruntergebrochen und die Chomsky-Normalform erreicht. —

$$P = \left\{ \begin{array}{l} S \rightarrow NS_N \mid ES_E \mid 0 \mid 1 \mid NN \mid EE \\ S_1 \rightarrow \varepsilon \mid S \\ S_N \rightarrow SN \\ S_E \rightarrow SE \\ N \rightarrow 0 \\ E \rightarrow 1 \end{array} \right\}$$

**66115 / 2017 / Frühjahr / Thema 2 / Aufgabe 2**

- (a) Gegeben sei die kontextfreie Grammatik  $G = (V, \Sigma, P, S)$  mit Sprache  $L(G)$ , wobei  $V = S, T, U$  und  $\Sigma = \{a, b, c, d, e\}$ .  $P$  bestehe aus den folgenden Produktionen:

$$P = \left\{ \begin{array}{l} S \rightarrow U \mid SbU \\ T \rightarrow dSe \mid a \\ U \rightarrow T \mid UcT \end{array} \right.$$



Der Automat auf flaci.com (FLACI: Formale Sprachen, abstrakte Automaten, Compiler und Interpreter) Ein Projekt der Hochschule Zittau/Görlitz und der Pädagogischen Hochschule Schwyz: flaci.com/Gib25c5oc

(i) Zeigen Sie  $acdae \in L(G)$ .

Lösungsvorschlag

$$S \vdash U \vdash UcT \vdash TcT \vdash acT \vdash acdSe \vdash acdUe \vdash acdae$$

(ii) Bringen Sie  $G$  in Chomsky-Normalform.

Lösungsvorschlag

**i. Elimination der  $\varepsilon$ -Regeln**

— Alle Regeln der Form  $A \rightarrow \varepsilon$  werden eliminiert. Die Ersetzung von  $A$  wird durch  $\varepsilon$  in allen anderen Regeln vorweggenommen.

$\emptyset$  Nichts zu tun

**ii. Elimination von Kettenregeln**

— Jede Produktion der Form  $A \rightarrow B$  mit  $A, B \in S$  wird als Kettenregel bezeichnet. Diese tragen nicht zur Produktion von Terminalzeichen bei und lassen sich ebenfalls eliminieren.

$$P = \left\{ \right.$$

$$S \rightarrow dSe \mid a \mid UcT \mid SbU$$

$$T \rightarrow dSe \mid a$$

$$U \rightarrow dSe \mid a \mid UcT$$

$$\left. \right\}$$

**iii. Separation von Terminalzeichen**

— Jedes Terminalzeichen  $\sigma$ , das in Kombination mit anderen Symbolen auftaucht, wird durch ein neues Nonterminal  $S_\sigma$  ersetzt und die Menge der Produktionen durch die Regel  $S_\sigma \rightarrow \sigma$  ergänzt.

$$P = \left\{ \right.$$

$$S \rightarrow DSE \mid a \mid UCT \mid SBU$$

$$T \rightarrow DSE \mid a$$

$$U \rightarrow DSE \mid a \mid UCT$$

$$B \rightarrow b$$

$$C \rightarrow c$$

$$D \rightarrow d$$

$$E \rightarrow e$$

$$\left. \right\}$$

**iv. Elimination von mehrelementigen Nonterminalketten**

— Alle Produktionen der Form  $A \rightarrow B_1B_2 \dots B_n$  werden in die Produktionen  $A \rightarrow A_{n-1}B_n, A_{n-1} \rightarrow A_{n-2}B_{n-1}, \dots, A_2 \rightarrow B_1B_2$  zerteilt. Nach der Ersetzung sind alle längeren Nonterminalketten vollständig heruntergebrochen und die Chomsky-Normalform erreicht.



$$P = \{$$

$$\begin{aligned} S &\rightarrow DS_E \mid a \mid UC_T \mid SB_U \\ T &\rightarrow DS_E \mid a \\ U &\rightarrow DS_E \mid a \mid UC_T \\ B &\rightarrow b \\ C &\rightarrow c \\ D &\rightarrow d \\ E &\rightarrow e \\ S_E &\rightarrow SE \\ C_T &\rightarrow CT \\ B_U &\rightarrow BU \end{aligned}$$

$$\}$$

- (b) Geben Sie eine kontextfreie Grammatik für  $L = \{a^i b^k c^i \mid i, k \in \mathbb{N} \mid a\}^n$ .

Lösungsvorschlag

Wir interpretieren  $\mathbb{N}$  als  $\mathbb{N}_0$ .

$$P = \{$$

$$S \rightarrow aSc \mid aBc \mid B \mid \varepsilon B \quad \rightarrow b \mid Bb$$

$$\}$$

Der Automat auf flaci.com (FLACI: Formale Sprachen, abstrakte Automaten, Compiler und Interpreter) Ein Projekt der Hochschule Zittau/Görlitz und der Pädagogischen Hochschule Schwyz: flaci.com/Ghp3bftdg

- (c) Zeigen Sie, dass  $L = \{a^i b^k c^i \mid i, k \in \mathbb{N} \wedge i < k \mid n\}$  nicht kontextfrei ist, indem Sie das Pumping-Lemma für kontextfreie Sprachen anwenden.

### Exkurs: Pumping-Lemma für Reguläre Sprachen

Es sei  $L$  eine kontextfreie Sprache. Dann gibt es eine Zahl  $j$ , sodass sich alle Wörter  $\omega \in L$  mit  $|\omega| \geq j$  zerlegen lassen in  $\omega = uvwxy$ , sodass die folgenden Eigenschaften erfüllt sind:

- (i)  $|vx| \geq 1$  (Die Wörter  $v$  und  $x$  sind nicht leer.)
- (ii)  $|vwx| \leq j$  (Die Wörter  $v$ ,  $w$  und  $x$  haben zusammen höchstens die Länge  $j$ .)
- (iii) Für alle  $i \in \mathbb{N}_0$  gilt  $uv^iwx^iy \in L$  (Für jede natürliche Zahl (mit 0)  $i$  ist das Wort  $uv^iwx^iy$  in der Sprache  $L$ )

**66115 / 2017 / Herbst / Thema 1 / Aufgabe 2**

Betrachten Sie die Sprache  $L_1 = L_a \cup L_b$ .

- $L_a = \{ a^n b c^n \mid n \in \mathbb{N} \}$
- $L_b = \{ a b^m c^m \mid m \in \mathbb{N} \}$

(a) Geben Sie für  $L_1$  eine kontextfreie Grammatik an.

Lösungsvorschlag

$$P = \left\{ \begin{array}{l} S \rightarrow S_a \mid S_b \\ S_a \rightarrow a S_a c \mid b \\ S_b \rightarrow a \mid a B_b \\ B_b \rightarrow b B_b c \mid bc \end{array} \right\}$$

(b) Ist Ihre Grammatik aus a) eindeutig? Begründen Sie Ihre Antwort.

Lösungsvorschlag

Nein. Die Sprache ist nicht eindeutig. Für das Wort  $abc$  gibt es zwei Ableitungen, nämlich  $S \vdash S_a \vdash a S_a c \vdash abc$  und  $S \vdash S_b \vdash a B_b \vdash abc$ .

(c) Betrachten Sie die Sprache  $L_2 = \{ a^{2^n} \mid n \in \mathbb{N} \}$ . Zeigen Sie, dass  $L_2$  nicht kontextfrei ist.

Lösungsvorschlag

Annahme:  $L_2$  ist kontextfrei  
 $\rightarrow$  Pumping-Lemma gilt für  $L_2$   
 $\rightarrow j \in \mathbb{N}$  als Pumping-Zahl  
 $\omega \in L_2: |\omega| \geq j$   
 Konsequenz:  $\omega = uvwxy$

- $|vx| \geq 1$
- $|vwx| \leq j$
- $uv^iwx^iy \in L_2$  für alle  $i \in \mathbb{N}_0$

Wir wählen:  $\omega = a^{2^i}: |\omega| \geq j$

**p**  $a \dots a$

**r**  $a \dots a$

**s**  $a \dots a$

**t**  $a \dots a$

**q**  $a \dots a$

$$q + r + s + t + q = 2^j$$

$$\Rightarrow r + t \geq 1$$

$$r + s + t \leq j$$

### 1. Fall

$$r + t = 2^{j-1}$$

$$2^{j-1} + 2^{j-1} = 2 \cdot 2^{j-1} = 2^1 \cdot 2^{j-1} = 2^{1+j-1} = 2^j$$

$$\omega' = uv^2wx^2y$$

$$p + 2 \cdot r + s + 2 \cdot t + q$$

$$p + s + q + 2 \cdot (r + t)$$

$$2^{j-1} + 2 \cdot 2^{j-1} = 3 \cdot 2^{j-1} = 2^{j-1} + 2^i \leq 2^{j+1}$$

keine Zweierpotenz

$$\Rightarrow \omega \notin L_2$$

$\Rightarrow$  Widerspruch zur Annahme

$\Rightarrow L_2$  nicht kontextfrei

### 2. Fall

$$r + t \neq 2^{j-1}$$

$$\omega' = uv^0wx^0y$$

$$\Rightarrow p + s + q = 2^j - (r + t)$$

$$(r + t) \neq 2^{j-i}$$

ist keine Zweierpotenz

$$\Rightarrow \omega \notin L_2$$

$\Rightarrow L_2$  nicht kontextfrei

## 66115 / 2017 / Herbst / Thema 2 / Aufgabe 5

Sei  $G = (\{S, A, B, C\}, \{a, b\}, P, S)$  die kontextfreie Grammatik in Chomsky-Normalform und der Menge  $P$  der Produktionen:

$$P = \left\{ \right.$$

$$\begin{aligned}
 S &\rightarrow AB \mid BC \\
 A &\rightarrow BA \mid a \\
 C &\rightarrow AB \mid a \\
 B &\rightarrow CC \mid b
 \end{aligned}$$

}

Sei  $\omega = baaab$ . Folgende Tabelle entsteht durch Anwendung des CYK-Algorithmus. Z. B. bedeutet  $B \in V(3,5)$ , dass aus der Variablen  $B$  das Teilwort  $\omega_3\omega_4\omega_5 = aab$  hergeleitet werden kann. Drei Einträge wurden weggelassen.

- (a) Bestimmen Sie die Mengen  $V(1,2)$ ,  $V(1,3)$  und  $V(1,5)$ .

Lösungsvorschlag

b	a	a	a	b
B	A,C	A,C	A,C	B
A,S	B	B	S,C	
-	S,C,A	B		
S,A,C	S,C			
S,C				

- (b) Wie entnehmen Sie dieser Tabelle, dass  $\omega \in L(G)$  ist?

Lösungsvorschlag

In der Menge  $V(1,5)$  ist das Startsymbol  $S$  der Sprache  $L(G)$  enthalten.

### 66115 / 2018 / Herbst / Thema 1 / Aufgabe 3

- (a) Entwerfen Sie eine kontextfreie Grammatik für die folgende kontextfreie Sprache über dem Alphabet  $\Sigma = \{a, b, c\}$ :

$$L = \{wb^{3k}c^{2k+1}v \mid k \in \mathbb{N}, |w|_c = |u|_a\}$$

(Hierbei bezeichnet  $|u|_x$  die Anzahl des Zeichens  $x$  in dem Wort  $u$ , und es gilt  $0 \in \mathbb{N}$ .) Erklären Sie den Zweck der einzelnen Nichtterminale (Variablen) und der Grammatikregeln Ihrer Grammatik.

- (b) Betrachten Sie die folgende kontextfreie Grammatik

$$G = (\{S, X, Y, Z\}, \{z, y\}, P, S)$$

mit den Produktionen

$$P = \left\{ \begin{array}{l} S \rightarrow ZX \mid y \\ X \rightarrow ZS \mid SS \mid x \\ Y \rightarrow SX \mid YZ \\ Z \rightarrow XX \mid XS \end{array} \right\}$$

Benutzen Sie den Algorithmus von Cocke-Younger-Kasami (CYK) um zu zeigen, dass das Wort  $xxxyx$  zu der von  $G$  erzeugten Sprache  $L(G)$  gehört.

Lösungsvorschlag

x	x	x	y	x
X	X	X	S	X
Z	Z	Z	Y	
S	X	S		
Z,X	Z			
X,S,Z				

$\Rightarrow xxxyx \in L(G)$

(c) Geben Sie eine Ableitung des Wortes  $xxxyx$  mit  $G$  an.

### 66115 / 2019 / Frühjahr / Thema 1 / Aufgabe 3

Gegeben sei die kontextfreie Grammatik  $G = (V, \Sigma, P, S)$  mit Sprache  $L(G)$ , wobei  $V = \{S, T, U\}$  und  $\Sigma = \{a, b\}$ .  $P$  bestehe aus den folgenden Produktionen:

$$P = \left\{ \begin{array}{l} S \rightarrow TUUT \\ T \rightarrow aT \mid \varepsilon \\ U \rightarrow bUb \mid a \end{array} \right\}$$

3

(a) Geben Sie fünf verschiedene Wörter  $w \in \Sigma^*$  mit  $w \in L(G)$  an.

<sup>3</sup><https://flaci.com/Gjpsin26a>

- aa
- aaaa
- ababbaba
- aababbabaa
- abbabbbabba

(b) Geben Sie eine explizite Beschreibung der Sprache  $L(G)$  an.

Lösungsvorschlag

$$L = \{ a^* b^n a b^{2n} a b^n a^* \mid n \in \mathbb{N}_0 \}$$

(c) Bringen Sie  $G$  in Chomsky-Normalform und erklären Sie Ihre Vorgehensweise.

Lösungsvorschlag

(i) **Elimination der  $\varepsilon$ -Regeln**

— Alle Regeln der Form  $A \rightarrow \varepsilon$  werden eliminiert. Die Ersetzung von  $A$  wird durch  $\varepsilon$  in allen anderen Regeln vorweggenommen. —

$$P = \{$$

$$S \rightarrow TUUT \mid TUU \mid UUT \mid UU$$

$$T \rightarrow aT \mid a$$

$$U \rightarrow bUb \mid a$$

$$\}$$

(ii) **Elimination von Kettenregeln**

— Jede Produktion der Form  $A \rightarrow B$  mit  $A, B \in S$  wird als Kettenregel bezeichnet. Diese tragen nicht zur Produktion von Terminalzeichen bei und lassen sich ebenfalls eliminieren. —

∅ Nichts zu tun

(iii) **Separation von Terminalzeichen**

— Jedes Terminalzeichen  $\sigma$ , das in Kombination mit anderen Symbolen auftaucht, wird durch ein neues Nonterminal  $S_\sigma$  ersetzt und die Menge der Produktionen durch die Regel  $S_\sigma \rightarrow \sigma$  ergänzt. —

$$P = \{$$

$$S \rightarrow TUUT \mid TUU \mid UUT \mid UU$$

$$T \rightarrow AT \mid A$$

$$U \rightarrow BUB \mid A$$

$$A \rightarrow a$$

$$B \rightarrow b$$

$$\}$$

(iv) **Elimination von mehrelementigen Nonterminalketten**

— Alle Produktionen der Form  $A \rightarrow B_1 B_2 \dots B_n$  werden in die Produktionen  $A \rightarrow A_{n-1} B_n, A_{n-1} \rightarrow A_{n-2} B_{n-1}, \dots, A_2 \rightarrow B_1 B_2$  zerteilt. Nach der Ersetzung sind alle längeren Nonterminalketten vollständig heruntergebrochen und die Chomsky-Normalform erreicht. —

$$P = \{$$

$$S \rightarrow TS_1 \mid TS_3 \mid US_2 \mid UU$$

$$S_1 \rightarrow US_2$$

$$S_2 \rightarrow UT$$

$$S_3 \rightarrow UU$$

$$T \rightarrow AT \mid a$$

$$U \rightarrow BU_1 \mid a$$

$$U_1 \rightarrow UB$$

$$A \rightarrow a$$

$$B \rightarrow b$$

$$\}$$
**66115 / 2020 / Frühjahr / Thema 1 / Aufgabe 3**

- (a) Entwerfen Sie eine kontextfreie Grammatik für die folgende kontextfreie Sprache über dem Alphabet  $\Sigma = \{a, b, c\}$ :

$$L = \{ a^{3n+2} w v c^n \mid n \in \mathbb{N}_0, 2 \cdot |w|_b = |v|_a \}$$

(Hierbei bezeichnet  $|u|_x$ , die Anzahl des Zeichens  $x$  in dem Wort  $u$ .)

Erklären Sie den Zweck der einzelnen Nichtterminale (Variablen) und der Grammatikregeln Ihrer Grammatik.

Lösungsvorschlag

$$P = \{$$

$$S \rightarrow aaaSc \mid aaaAc$$

$$A \rightarrow aaB$$

$$B \rightarrow bBaa \mid baa$$

$$\}$$

Der Automat auf flaci.com (FLACI: Formale Sprachen, abstrakte Automaten, Compiler und Interpreter) Ein Projekt der Hochschule Zittau/Görlitz und der Pädagogischen Hochschule Schwyz: flaci.com/Ghhs1xexw

(b) Betrachten Sie die folgende kontextfreie Grammatik

$$G = (\{A, B, C, D\}, \{a, b, c\}, P, A)$$

mit den Produktionen

$$P = \left\{ \begin{array}{l} A \rightarrow AB \mid CD \mid a \\ B \rightarrow CC \mid c \\ C \rightarrow DC \mid CB \mid b \\ D \rightarrow DB \mid a \end{array} \right\}$$

Der Automat auf flaci.com (FLACI: Formale Sprachen, abstrakte Automaten, Compiler und Interpreter) Ein Projekt der Hochschule Zittau/Görlitz und der Pädagogischen Hochschule Schwyz: flaci.com/Gf7556jn2

Benutzen Sie den Algorithmus von Cocke-Younger-Kasami (CYK), um zu zeigen, dass das Wort *abcab* zu der von  $G$  erzeugten Sprache  $L(G)$  gehört.

Lösungsvorschlag

a	b	c	a	b
A,D	C	B	A,D	C
C	C	-	C	
C,C	A	-		
A,A	B			
A,D,B,B				

$\Rightarrow abcab \in L(G)$

(c) Finden Sie nun ein größtmögliches Teilwort von *abcab*, dass von keinem der vier Nichtterminale von  $G$  ableitbar ist.

(d) Geben Sie eine Ableitung des Wortes *abcab* mit  $G$  an.

Lösungsvorschlag

$$A \vdash AB \vdash ACC \vdash ACBC \vdash ACBDC \vdash aCBDC \vdash abBDC \vdash abcDC \vdash abcaC \vdash abcab$$

(e) Beweisen Sie, dass die folgende formale Sprache über  $Z = a, b$  nicht kontextfrei ist:  $L = \{a^n b^n \mid n \in \mathbb{N}\}$ .



**66115 / 2020 / Frühjahr / Thema 2 / Aufgabe 3**

- (a) Ist die folgende Sprache  $L_1 = \{a^{n+2}b^{2n+1} \mid n \geq 2\}$  über dem Alphabet  $\Sigma = \{a, b\}$  kontextfrei?

Falls ja, geben Sie eine kontextfreie Grammatik für  $L_1$ , an, falls nein, eine kurze Begründung (ein vollständiger Beweis ist hier nicht gefordert).

Lösungsvorschlag

$L_1$  ist kontextfrei

$G = (\{S, A, B\}, \{a, b\}, P, S)$

$P = \{$

$S \rightarrow aAbb$

$A \rightarrow aAbb \mid aBbb$

$B \rightarrow aab$

$\}$

Der Automat auf flaci.com (FLACI: Formale Sprachen, abstrakte Automaten, Compiler und Interpreter) Ein Projekt der Hochschule Zittau/Görlitz und der Pädagogischen Hochschule Schwyz: flaci.com/Grxk1oczg

$n = 2$  4a 5b: aaaabbbbb

$n = 3$  5a 7b: aaaaabbbbbbb

$n = 4$  6a 9b: aaaaaabbbbbbbb

- (b) Geben Sie einen Kellerautomaten (PDA) formal an, der die Sprache

$L_1 = \{w_1w_2w_3 \mid w_1, w_2, w_3 \in \Sigma^* \setminus \{\lambda\} \text{ und } w_1 = w_3^{\text{rev}}\} \in CFL$  über dem Alphabet  $\Sigma = \{0, 1\}$  akzeptiert.

Dabei bezeichnet  $\lambda$  das leere Wort und  $w_3^{\text{rev}}$  bezeichnet das Wort  $w_3$  rückwärts gelesen. Bei Akzeptanz einer Eingabe soll sich der PDA in einem Endzustand befinden und der Keller geleert sein.

Lösungsvorschlag

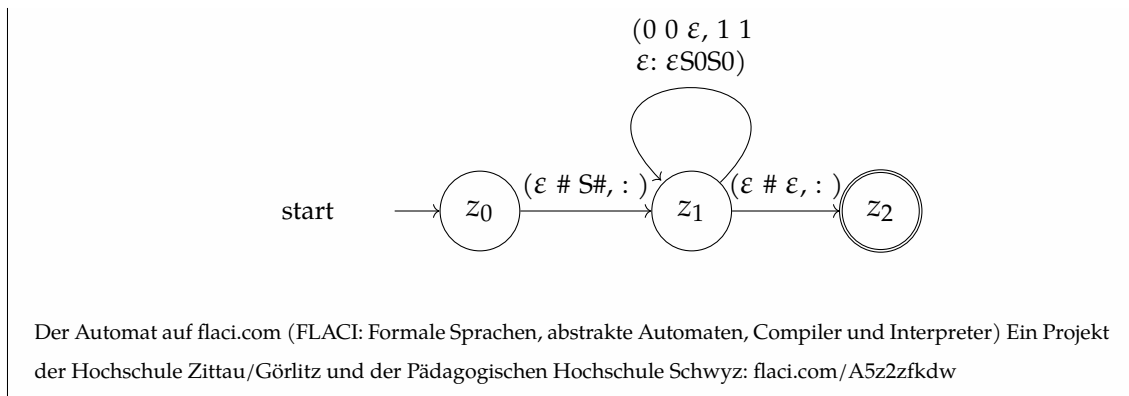
Der Automat auf flaci.com (FLACI: Formale Sprachen, abstrakte Automaten, Compiler und Interpreter) Ein Projekt der Hochschule Zittau/Görlitz und der Pädagogischen Hochschule Schwyz: flaci.com/Gpkctmk3g

$P = \{$

$S \rightarrow 0S0 \mid 1S1 \mid 0A0 \mid 1A1$

$A \rightarrow 0A \mid 1A \mid 0 \mid 1$

$\}$



(c) Beschreiben Sie in Worten die Arbeitsweise Ihres PDA aus Aufgabenteil (b).

## 66115 / 2020 / Herbst / Thema 2 / Teilaufgabe 1 / Aufgabe 2

- (a) Sei  $L = \{ 0^n 1^m 1^p 0^q \mid n + m = p + q \text{ und } n, m, p, q \in \mathbb{N}_0 \}$ . Geben Sie eine kontextfreie Grammatik für  $L$  an. Sie dürfen dabei  $\varepsilon$ -Produktionen der Form  $\{A \rightarrow \varepsilon\}$  verwenden.

Lösungsvorschlag

$$P = \left\{ \begin{array}{l} S \rightarrow 0S0 \mid 0A0 \mid 0B0 \mid \varepsilon \mid A \mid B \mid C \\ A \rightarrow 0A1 \mid 0C1 \\ B \rightarrow 1B0 \mid 1C0 \\ C \rightarrow 1C1 \mid \varepsilon \end{array} \right\}$$

- (b) Für eine Sprache  $L$  sei  $L^r = \{ x^r \mid x \in L \}$  die Umkehrsprache. Dabei bezeichne  $x^r$  das Wort, das aus  $x$  entsteht, indem man die Reihenfolge der Zeichen umkehrt, beispielsweise  $(abb)^r = bba$ .
- (i) Sei  $L$  eine kontextfreie Sprache. Zeigen Sie, dass dann auch  $L^r$  kontextfrei ist.
  - (ii) Geben Sie eine kontextfreie Sprache  $L_1$ , an, sodass  $L_1 \cap L_1^r$  kontextfrei ist.
  - (iii) Geben Sie eine kontextfreie Sprache  $L_2$ , an, sodass  $L_2 \cap L_2^r$  nicht kontextfrei ist.

## 66115 / 2021 / Frühjahr / Thema 1 / Teilaufgabe 1 / Aufgabe 2

Sei  $G = (V, \Sigma, P, S)$  eine kontextfreie Grammatik mit Variablen  $V = \{S, A, B, C, D\}$ , Terminalzeichen  $\Sigma = \{a, b, c\}$ , Produktionen

$$P = \left\{ \right.$$

$$S \rightarrow AD \mid CC \mid c$$

$$A \rightarrow a$$

$$B \rightarrow b$$

$$C \rightarrow CC \mid c$$

$$D \rightarrow SB \mid CB$$

}

und Startsymbol  $S$ . Führen Sie den Algorithmus von Cocke, Younger und Kasami (CYK-Algorithmus) für  $G$  und das Wort  $aaaccbbb$  aus. Liegt  $aaaccbbb$  in der durch  $G$  erzeugten Sprache? Erläutern Sie Ihr Vorgehen und den Ablauf des CYK-Algorithmus.

Lösungsvorschlag

a	a	a	c	c	b	b	b
-	-	-	S,C	D,D	-	-	
-	-	-	D,D	-	-		
-	-	S,S	-	-			
-	-	D,D	-				
-	S,S	-					
-	D,D						
S,S							

Das Wort  $aaaccbbb$  liegt in der Sprache.

## 66115 / 2021 / Frühjahr / Thema 2 / Teilaufgabe 1 / Aufgabe 2

(a) Zeigen Sie, dass die Sprache

$$L = \{ ww_1ww_2 \mid w, w_1, w_2 \in \{a, b, c\}^* \text{ und } 2|w| \geq |w_1| + |w_2| \}$$

nicht kontextfrei ist.

### Exkurs: Pumping-Lemma für Kontextfreie Sprachen

Es sei  $L$  eine kontextfreie Sprache. Dann gibt es eine Zahl  $j$ , sodass sich alle Wörter  $\omega \in L$  mit  $|\omega| \geq j$  zerlegen lassen in  $\omega = uvwxy$ , sodass die folgenden Eigenschaften erfüllt sind:

- (i)  $|vx| \geq 1$  (Die Wörter  $v$  und  $x$  sind nicht leer.)
- (ii)  $|vwx| \leq j$  (Die Wörter  $v$ ,  $w$  und  $x$  haben zusammen höchstens die Länge  $j$ .)
- (iii) Für alle  $i \in \mathbb{N}_0$  gilt  $uv^iwx^iy \in L$  (Für jede natürliche Zahl (mit 0)  $i$  ist das Wort  $uv^iwx^iy$  in der Sprache  $L$ )

Es gibt eine Pumpzahl. Sie sei  $j$ .  $a^j b^j a^j c^j$  ist ein Wort aus  $L$ , das sicher länger als  $j$  ist. Außerdem gilt  $2|a^j| \geq |b^j| + |c^j|$ . Unser gewähltes Wort ist deshalb in  $L$ .

Da  $|vwx| \leq j$  und  $|xv| \geq 1$  sein muss, liegt  $vwx$  entweder in  $w$ ,  $w_1$  oder  $w_2$ .

**Aufteilung:  $vwx$  in  $w$  (erstes  $w$ ):**

**u** :  $\varepsilon$

**v** :  $a$

**w** :  $a^{j-2}$

**x** :  $a$

**y** :  $b^j a^j c^j$

Es gilt  $uv^i wx^i y \notin L$  für alle  $i \in \mathbb{N}_0$ , da  $a^j b^j a^j c^j \notin L$  für  $i = 0$ , da  $|a^{j-2}| + |a^j| < |b^j| + |c^j|$

**Aufteilung:  $vwx$  in  $w$  (zweites  $w$ ):**

**u** :  $a^j b^j$

**v** :  $a$

**w** :  $a^{j-2}$

**x** :  $a$

**y** :  $c^j$

Es gilt  $uv^i wx^i y \notin L$  für alle  $i \in \mathbb{N}_0$ , da  $a^j b^j a^j c^j \notin L$  für  $i = 0$ , da  $|a^j| + |a^{j-2}| < |b^j| + |c^j|$

**Aufteilung:  $vwx$  in  $w_1$ :**

**u** :  $a^j$

**v** :  $b$

**w** :  $b^{j-2}$

**x** :  $b$

**y** :  $a^j c^j$

Es gilt nicht  $uv^i wx^i y \in L$  für alle  $i \in \mathbb{N}_0$ , da  $a^j b^j a^j c^j \notin L$  für alle  $i > 2$  da  $2|a^j| < |b^{j-2+2i}| + |c^j|$  für alle  $i > 2$

**Aufteilung:  $vwx$  in  $w_2$ :**

Analog zur Aufteilung  $vwx$  in  $w_1$

$\Rightarrow L$  ist nicht kontextfrei.

(b) Betrachten Sie die Aussage

Seien  $L_1, \dots, L_n$  beliebige kontextfreie Sprachen.  
Dann ist  $\bigcap_{i=1}^n L_i$  immer eine entscheidbare Sprache.

Entscheiden Sie, ob diese Aussage wahr ist oder nicht und begründen Sie Ihre Antwort.

Lösungsvorschlag

Diese Aussage ist falsch.

Kontextfreie Sprachen sind nicht abgeschlossen unter dem Schnitt, die Schnittmenge zweier kontextfreier Sprachen kann in einer Sprache eines anderen Typs in der Chomsky Sprachen-Hierarchie resultieren. Entsteht durch den Schnitt eine Typ-0-Sprache, dann ist diese nicht entscheidbar.

(c) Sei  $\mathbb{N}_0 = \{0, 1, 2, \dots\}$  die Menge der nicht negativen natürlichen Zahlen. Es ist bekannt, dass  $L = \{a^n b^n c^n \mid n \in \mathbb{N}\}$  keine kontextfreie Sprache ist. Ist die Komplementsprache  $L_5 = \{a, b, c\}^* \setminus L$  kontextfrei? Begründen Sie Ihre Antwort.

# Kontextsensitive Sprache

66115 / 2021 / Frühjahr / Thema 2 / Aufgabe 2

## Kontextsensitive Grammatik

Sei  $L = \{ a^n b^m c^n \mid n, m \geq 0, m \leq n \}$ .

(a) Geben Sie eine kontextsensitive Grammatik für  $L$  an.

Lösungsvorschlag

$G = (\{S, A, B, C\}, \{a, b, c\}, P, S)$  mit Produktionen  $P$  wie folgt:

$P = \{$

$S \rightarrow \varepsilon \mid aSc \mid aSBC$   
 $CB \rightarrow BC$   
 $bC \rightarrow bc$   
 $aB \rightarrow ab$   
 $bB \rightarrow bb$   
 $cC \rightarrow cc$

$\}$

(b) Geben Sie eine Ableitung des Wortes  $a^3 b^2 c^3$  mittels der in Teilaufgabe a) erstellten Grammatik an.

Lösungsvorschlag

$S \vdash aSc \vdash aaSBCc \vdash aaaSBCBCc \vdash aaaBCBCc \vdash aaaBBCCc \vdash aaabBCCc \vdash aaabbCCc \vdash aaabbcCc \vdash aaabbccc$

66115 / 2021 / Frühjahr / Thema 2 / Aufgabe 2

## Übung kontextsensitive Grammatiken [Seite 8]theo:fs:3

Gegen sei folgende Grammatik:

$G = (V, \Sigma, P, S)$  mit  $V = \{S, B, C\}$ ,  $\Sigma = \{a, b, c\}$ ,  $S = S$  und

$P = \{$

$$S \rightarrow aSBC \mid aBC$$

$$CB \rightarrow BC$$

$$aB \rightarrow ab$$

$$bB \rightarrow bb$$

$$bC \rightarrow bc$$

$$cC \rightarrow cc$$

}

- (a) Geben Sie die Sprache an, die die folgende Grammatik erzeugt:

Lösungsvorschlag

$$L = \{ a^n b^b c^n \mid n \in \mathbb{N} \}$$

- (b) Gib Sie eine Ableitung mit der folgenden Grammatik für das Wort aaabbbccc an.

Lösungsvorschlag

$aSBC \vdash aaSBCBC \vdash aaaBCBCBC \vdash aaaBBCCBC \vdash aaaBBCBCC \vdash aaaBBBCCC \vdash$   
 $aaabBBCCC \vdash aaabbBCCC \vdash aaabbbCCC \vdash aaabbbcCC \vdash aaabbbccC \vdash aaabbbccc$

## 66115 / 2021 / Frühjahr / Thema 2 / Aufgabe 2

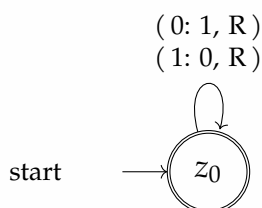
### Komplement der Binärzahl

### Übung zu Turingmaschinen

Gegeben ist eine Binärzahl auf dem Band einer Turingmaschine.

- (a) Definieren Sie vollständig eine TM, die das Komplement der Binärzahl (0110  $\rightarrow$  1001) berechnet. Die Überföhrungsfunktion kann als Tabelle oder als Graph angegeben werden.

Lösungsvorschlag



Der Automat auf flaci.com (FLACI: Formale Sprachen, abstrakte Automaten, Compiler und Interpreter) Ein Projekt der Hochschule Zittau/Görlitz und der Pädagogischen Hochschule Schwyz: flaci.com/Ap9qtjgg7

name: Komplement Binärzahl

init: z0

accept: z0

z0,0

$z_0, 1, >$

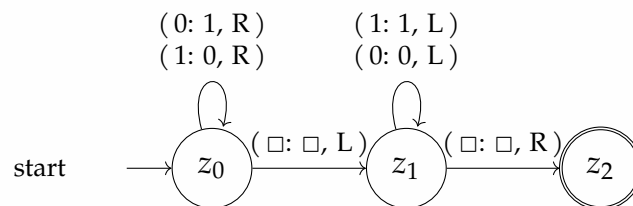
$z_0, 1$   
 $z_0, 0, >$

*a*

<sup>a</sup><http://turingmachinesimulator.com/shared/lqpsawxdeh>

- (b) Erweitern Sie Ihre Maschine aus Aufgabe a) so, dass der Schreib-/Lesekopf auf dem ersten Zeichen der Eingabe terminiert.

Lösungsvorschlag



Der Automat auf [flaci.com](http://flaci.com) (FLACI: Formale Sprachen, abstrakte Automaten, Compiler und Interpreter) Ein Projekt der Hochschule Zittau/Görlitz und der Pädagogischen Hochschule Schwyz: [flaci.com/A5o7tug5r](http://flaci.com/A5o7tug5r)



# Unbeschränkte Sprache

66115 / 2021 / Frühjahr / Thema 2 / Aufgabe 2

## Binärzahl dekrementieren

Sei  $\Sigma = \{0, 1\}$  und  $\Gamma = \{0, 1, \square\}$ . Konstruiere eine Turingmaschine  $M$ , die eine in Binärform gegebene, natürliche Zahl ( $\neq 0$ ) um 1 dekrementiert (und wieder in Binärform ausgibt). Der Schreib-/Lesekopf steht zu Beginn der Berechnung auf dem ersten Leerzeichen links von der Eingabe und soll auch am Ende wieder dort stehen. Beachte, dass führende Nullen in der Eingabe/Ausgabe nicht vorkommen dürfen.

Lösungsvorschlag

dezimal	binär
0	0
1	1
2	10
3	11
4	100
5	101
6	110
7	111
8	1000
9	1001
10	1010
11	1011
12	1100
13	1101
14	1110
15	1111
16	10000

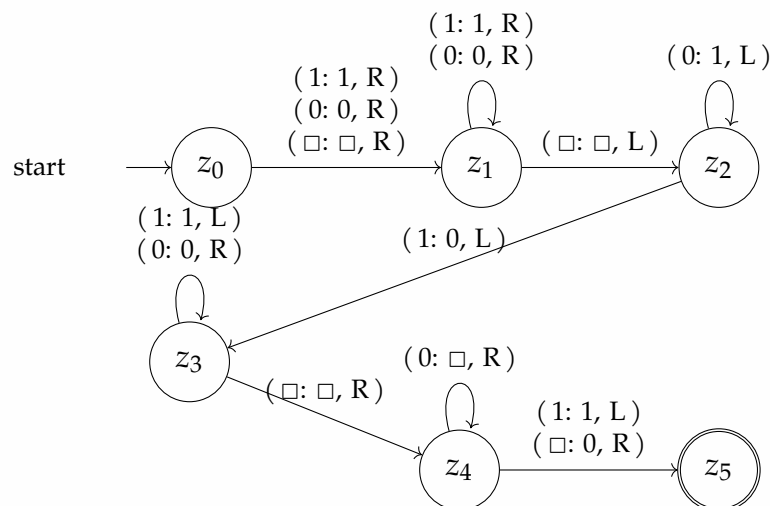
Die Maschine geht zunächst ans rechte Ende des Wortes, dann invertiert sie alle 0 Bits, bis sie auf eine 1 trifft. Diese wird durch 0 ersetzt. Damit ist der Dekrementierungsvorgang beendet. Nun sucht Sie das linke Ende des Wortes und löscht eventuell entstandene führende Nullen. Trifft Sie dabei auf das Leerzeichen, so war die

Abgabe die Zahl 0 und diese wird wieder aufs Band geschrieben. Insgesamt ergibt sich

$$TM = (\{z_0, z_1, z_2, z_3, z_4, z_5\}, \Sigma, \Gamma, \delta, z_0, \square, \{z_5\})$$

mit unten angegebener Übergangsfunktion:

$\delta$	0	1	d	Kommentar
$z_0$	$\emptyset$	$\emptyset$	$(z_1: \square, R)$	Gehe auf erstes Zeichen des Wortes
$z_1$	$(z_1: 0, R)$	$(z_1: 1, R)$	$(z_2: \square, L)$	Gehe ans rechte Ende des Wortes
$z_2$	$(z_2: 1, L)$	$(z_3: 0, L)$	$\emptyset$	Flippe alle 0 Bits bis die erste 1 erreicht wird, setze die
$z_3$	$(z_3: 0, L)$	$(z_3: 1, L)$	$(z_4: \square, R)$	suche linkes Ende des Wortes
$z_4$	$(z_4: \square, R)$	$(z_5: 1, L)$	$(z_5: 0, L)$	lösche führende Nullen , schreibe evtl. 0 aufs Band
$z_5$	$\emptyset$	$\emptyset$	$\emptyset$	Endzustand



Der Automat auf [flaci.com](http://flaci.com) (FLACI: Formale Sprachen, abstrakte Automaten, Compiler und Interpreter) Ein Projekt der Hochschule Zittau/Görlitz und der Pädagogischen Hochschule Schwyz: [flaci.com/Ahifz611c](http://flaci.com/Ahifz611c)

## 66115 / 2021 / Frühjahr / Thema 2 / Aufgabe 2

### Turing-Maschine zur Multiplikation mit 3

Gesucht ist eine Turing-Maschine, die die Funktion  $f: \mathbb{N} \rightarrow \mathbb{N}$  mit  $f(x) = 3x$  berechnet. Zu Beginn der Berechnung steht die Eingabe binär codiert auf dem Band, wobei der Kopf auf die linkeste Ziffer (most significant bit) zeigt. Am Ende der Berechnung soll der Funktionswert binär codiert auf dem Band stehen, wobei der Kopf auf ein beliebiges Feld zeigen darf.

- (a) Überlege, was bei der Multiplikation mit 3 im Binären tatsächlich passiert. Leite hieraus die Arbeitsweise des Algorithmus für die Turingmaschine ab und be-

schreibe diese. Tipp: Die schriftliche Multiplikation mit Binärzahlen funktioniert genauso wie die schriftliche Multiplikation mit Dezimalzahlen!

Lösungsvorschlag

$$13 \cdot 3 = 0b1101 \cdot 0b11 = 39 = 0b100111:$$

$$\begin{array}{r}
 1\ 1\ 0\ 1 \\
 1\ 1\ 0\ 1 \\
 \hline
 1\ 0\ 0\ 1\ 1\ 1
 \end{array}$$

- (b) Erstelle dazu eine TM mit 3 Bändern. Die ersten beiden Bänder sollen für die Berechnung herangezogen werden. Auf dem dritten Band soll das Ergebnis stehen. (analog zur schriftlichen Multiplikation)

Lösungsvorschlag

$z_0$  Versetzung der Zeiger von Band 2 und 3 im Vergleich zu 1.  
 $z_1$  kopiert das Wort von Band 1 auf Band 2 (mit Versetzung).  
 $z_2$  Binäre Addition ohne Übertrag.  
 $z_3$  Binäre Addition mit Übertrag.  
 $z_4$  Übertrag am Ende der Addition falls sich das Wort vergrößert  
 $z_f$  fertig

name: Multiplikation mit 3 (3-Band)

init:  $z_0$

accept:  $z_f$

$z_0$ , 0, \_ , \_

$z_1$ , 0, \_ , \_ , - , > , >

$z_0$ , 1, \_ , \_

$z_1$ , 1, \_ , \_ , - , > , >

$z_1$ , 0, \_ , \_

$z_1$ , 0, 0, \_ , > , > , >

$z_1$ , 1, \_ , \_

$z_1$ , 1, 1, \_ , > , > , >

$z_1$ , \_ , \_ , \_

$z_2$ , \_ , \_ , \_ , - , < , <

$z_2$ , \_ , 0, \_

$z_2$ , \_ , 0, 0, < , < , <

$z_2$ , \_ , 1, \_

$z_2$ , \_ , 1, 1, < , < , <

$z_2$ , 0, 0, \_

$z_2$ , 0, 0, 0, < , < , <

```

z2, 0,1,_
z2, 0,1,1, <,<,<
z2, 1,0,_
z2, 1,0,1, <,<,<

```

```

z2, 1,_,-
zf, 1,_,-,1, -,-,-

```

```

z2, 1,1,_
z3, 1,1,0, <,<,<

```

```

z3, 0,1,_
z3, 0,1,0, <,<,<
z3, 1,0,_
z3, 1,0,0, <,<,<
z3, 1,1,_
z3, 1,1,1, <,<,<

```

```

z3, 0,0,_
z2, 0,0,1, <,<,<

```

```

z3, 1,_,-
z4, 1,_,-,0, <,<,<

```

```

z4, _,-,-
zf, _,-,-,1, -,-,-

```

*a*

<sup>a</sup><http://turingmachinesimulator.com/shared/tgaybidewo>

- (c) Erstelle dazu eine TM mit 2 Bändern. Auf dem ersten Band steht die Eingabe und auf dem zweiten Band soll das Ergebnis stehen.

Lösungsvorschlag

```

z0  Die Binärzahl überlaufen
z1  Zur nächsten Ziffer muss 0 addiert werden.
z2  Zur nächsten Ziffer muss 1 addiert werden.
z3  Zur nächsten Ziffer muss 2 addiert werden.
zf  fertig

```

```

name: Multiplikation mit 3 (2-Band)
init: z0
accept: zf

```

```

z0, 0,_
z0, 0,_,-, >,-
z0, 1,_
z0, 1,_,-, >,-
z0, _,-,-

```

$z_1, \_ , \_ , < , -$

$z_1, 0, \_$

$z_1, 0, 0, < , <$

$z_1, 1, \_$

$z_2, 1, 1, < , <$

$z_2, 1, \_$

$z_3, 1, 0, < , <$

$z_3, 0, \_$

$z_2, 0, 0, < , <$

$z_3, \_ , \_$

$z_2, \_ , 0, - , <$

$z_2, 0, \_$

$z_1, 0, 1, < , <$

$z_2, \_ , \_$

$z_1, \_ , 1, - , <$

$z_3, 1, \_$

$z_3, 1, 1, < , <$

$z_1, \_ , \_$

$z_f, \_ , \_ , - , -$

$a$

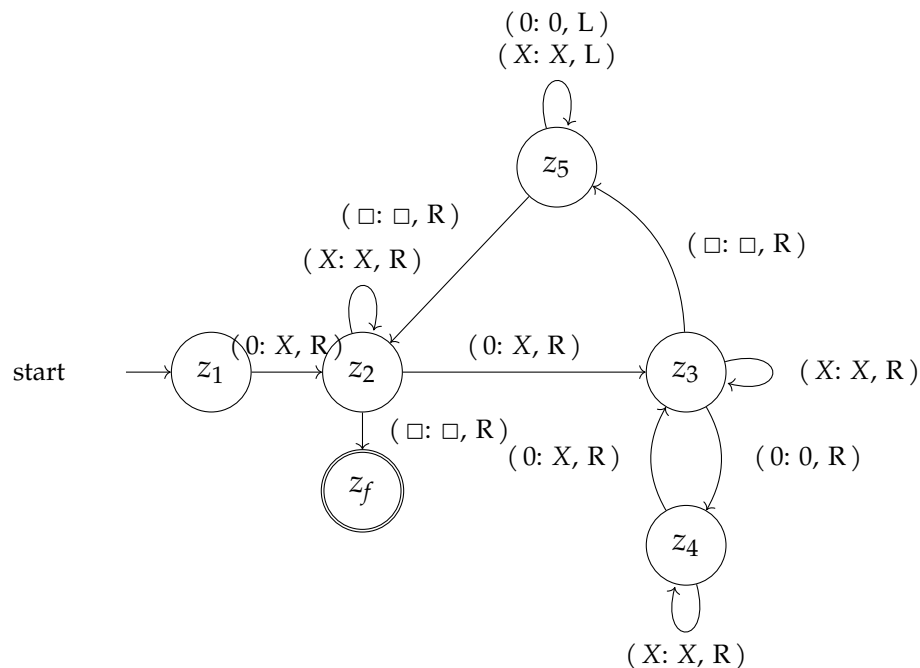
<sup>a</sup><http://turingmachinesimulator.com/shared/caoxpsgrgl>

## 66115 / 2021 / Frühjahr / Thema 2 / Aufgabe 2

### Turingmaschine mit folgender Übergangsfunktion

Gegeben sei eine TM mit folgender Übergangsfunktion:

	$z_1$	$z_2$	$z_3$	$z_4$	$z_5$
0	$(z_2: \square, R)$	$(z_3: X, R)$	$(z_4: 0, R)$	$(z_3: X, R)$	$(z_5: 0, L)$
X	-	$(z_2: X, R)$	$(z_3: X, R)$	$(z_4: X, R)$	$(z_5: X, L)$
$\square$	-	$(z_f: \square, R)$	$(z_5: \square, L)$	-	$(z_2: \square, R)$



Der Automat auf flaci.com (FLACI: Formale Sprachen, abstrakte Automaten, Compiler und Interpreter) Ein Projekt der Hochschule Zittau/Görlitz und der Pädagogischen Hochschule Schwyz: [flaci.com/Apew8cea2](http://flaci.com/Apew8cea2)

Erreicht die TM den Zustand  $z_f$  (final), so hält sie an und bearbeitet keine weitere Eingabe. Zu Beginn der Berechnung soll die TM auf dem ersten Symbol der Eingabe (links) stehen.

(a) Gebe für die folgenden Eingaben die Konfigurationsfolgen der Berechnung an:

- 00000

Lösungsvorschlag

Der Zustand der TM steht vor dem nächsten gelesenen Zeichen

$z_1$  00000  $\rightarrow$   
 $\rightarrow$   $\square$   $z_2$  0000  
 $\rightarrow$   $\square$  X  $z_3$  000  
 $\rightarrow$   $\square$  X0  $z_4$  00  
 $\rightarrow$   $\square$  X0X  $z_3$  0  
 $\rightarrow$   $\square$  X0X0  $z_4$

- 000000

Lösungsvorschlag

Der Zustand der TM steht vor dem nächsten gelesenen Zeichen

$z_1$  000000  $\rightarrow$   
 $\rightarrow \square z_2$  00000  
 $\rightarrow \square X z_3$  0000  
 $\rightarrow \square X0 z_4$  000  
 $\rightarrow \square X0X z_3$  00  
 $\rightarrow \square X0X0 z_4$  0  
 $\rightarrow \square X0X0X z_3$   $\square$   
 $\rightarrow \square X0X0 z_5$   $X \square$   
 $\rightarrow \square X0X z_5$   $0X \square$   
 $\rightarrow \square X0 z_5$   $X0X \square$   
 $\rightarrow \square X z_5$   $0X0X \square$   
 $\rightarrow \square z_5$   $X0X0X \square$   
 $\rightarrow z_5 \square X0X0X \square$   
 $\rightarrow \square z_2$   $X0X0X \square$   
 $\rightarrow \square X z_2$   $0X0X \square$   
 $\rightarrow \square XX z_3$   $X0X \square$   
 $\rightarrow \square XXX z_3$   $0X \square$   
 $\rightarrow \square XXX0 z_4$   $X \square$   
 $\rightarrow \square XXX0X z_4$   $\square$

- 0000

Lösungsvorschlag

Der Zustand der TM steht vor dem nächsten gelesenen Zeichen

$z_1 000 \rightarrow$   
 $\rightarrow \square z_1 000$   
 $\rightarrow \square X z_3 00$   
 $\rightarrow \square X0 z_4 0$   
 $\rightarrow \square X0X z_3 \square$   
 $\rightarrow \square X0 z_5 X \square$   
 $\rightarrow \square X z_5 0X \square$   
 $\rightarrow \square z_5 X0X \square$   
 $\rightarrow z_5 \square X0X \square$   
 $\rightarrow \square z_2 X0X \square$   
 $\rightarrow \square X z_2 0X \square$   
 $\rightarrow \square XX z_3 X \square$   
 $\rightarrow \square XXX z_3 \square$   
 $\rightarrow \square XX z_5 X \square$   
 $\rightarrow \square X z_5 XX \square$   
 $\rightarrow \square z_5 XXX \square$   
 $\rightarrow z_5 \square XXX \square$   
 $\rightarrow \square z_2 XXX \square$   
 $\rightarrow \square X z_2 XX \square$   
 $\rightarrow \square XX z_2 X \square$   
 $\rightarrow \square XXX z_2 \square$   
 $\rightarrow \square XXX \square z_f$

- (b) Gebe zwei andere Wörter über der Sprache  $L \subset \{0^*\}$  an, für die TM im Zustand  $z_f$  endet.

Lösungsvorschlag

z. B. 0 oder 00

- (c) Für welche Sprache ist die TM ein Akzeptor?

Lösungsvorschlag

Die TM erkennt alle Wörter mit der Eigenschaft, dass die Anzahl der Nullen eine 2er-Potenzen ist.



## 66115 / 2021 / Frühjahr / Thema 2 / Aufgabe 2

## Wörter umkehren auf der 2-Band-Turingmaschine

- (a) Geben Sie eine 2-Band-Turingmaschine an, die die Eingabe über dem Alphabet  $\Sigma = \{a, b\}$  umkehrt.

Beispiele:

- $abb \rightarrow bba$
- $aaaaba \rightarrow abaana$
- $aaa \rightarrow aaa$

Tipp: Das Ergebniswort muss nicht auf dem 1. Band stehen.

Lösungsvorschlag

```

name: ab-Wörter umkehren 2-Band-Turingmaschine
init: z0
accept: z0

z0,a,_
z0,_,a,>,<

z0,b,_
z0,_,b,>,<

a
_____
ahttp://turingmachinesimulator.com/shared/iqwxphngwc

```

## 66115 / 2021 / Frühjahr / Thema 2 / Aufgabe 2

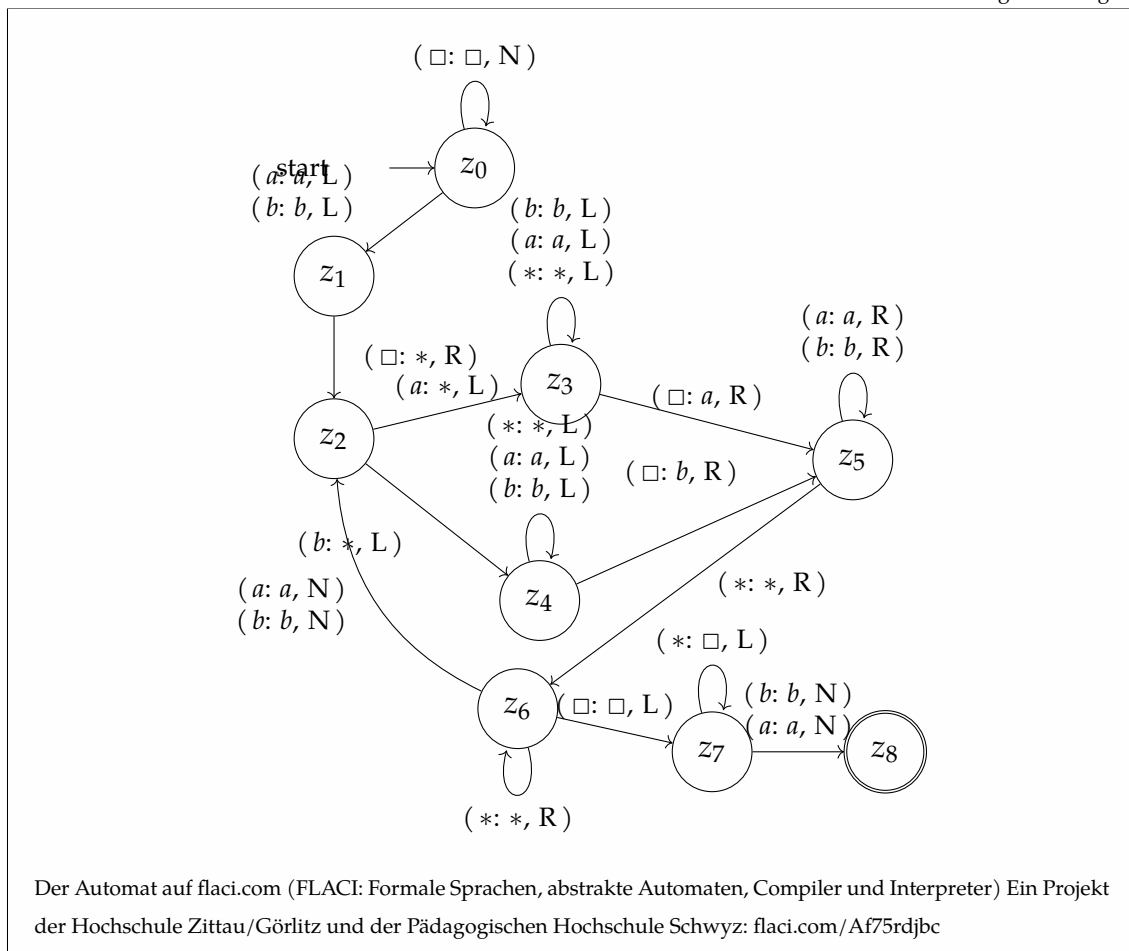
## Wörter umkehren auf dem Band der Turingmaschine

- (a) Geben Sie eine Turingmaschine an, die die Eingabe über dem Alphabet  $\Sigma = \{a, b\}$  umkehrt.

Beispiele:

- $abb \rightarrow bba$
- $aaaaba \rightarrow abaana$
- $aaa \rightarrow aaa$

Tipp: Fügen Sie ein extra Zeichen ein, welches das Eingabewort von deinem umgedrehten Wort trennt. Das Ergebniswort muss nicht an derselben Stelle wie das Eingabewort stehen.



(b) Geben Sie anschließend eine Konfigurationsfolge ihrer TM für  $ab$  an.

$$\begin{aligned}
z_0 ab &\rightarrow z_1 \square ab \\
&\rightarrow z_2 * ab \\
&\rightarrow * z_3 * b \\
&\rightarrow z_3 * * b \\
&\rightarrow z_3 \square * * b \\
&\rightarrow a z_5 * * b \\
&\rightarrow a * z_6 * b \\
&\rightarrow a * * z_6 b \\
&\rightarrow a * * z_2 b \\
&\rightarrow a * z_4 * * \\
&\rightarrow a z_4 * * * \\
&\rightarrow z_4 a * * * \\
&\rightarrow z_4 \square a * * * \\
&\rightarrow b z_5 a * * * \\
&\rightarrow ba z_6 * * * \\
&\rightarrow ba * z_6 * * \\
&\rightarrow ba * * z_6 * \\
&\rightarrow ba * * * z_6 \square \\
&\rightarrow ba * * z_7 \square \\
&\rightarrow ba * z_7 \square \\
&\rightarrow ba z_7 \square \\
&\rightarrow b z_8 a
\end{aligned}$$

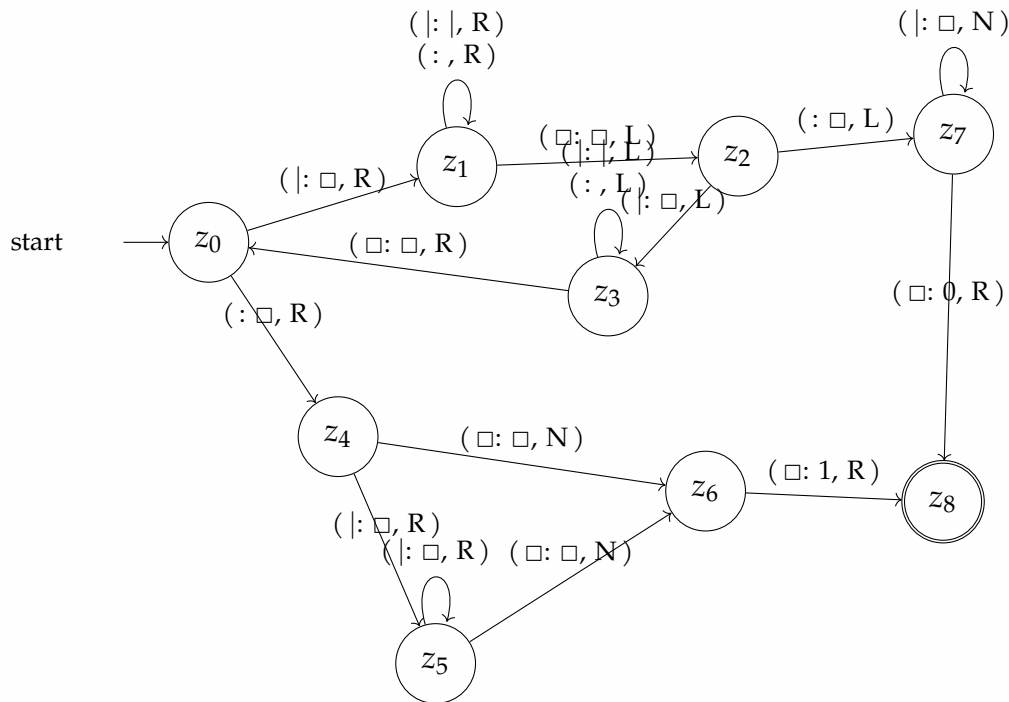
## 66115 / 2021 / Frühjahr / Thema 2 / Aufgabe 2

### Prädikat unäre Kodierung von $n$ und $m$

Konstruieren Sie eine Einband-Turingmaschine, die für eine Eingabe der Form  $|^n \circ |^m$  mit  $n, m \in \mathbb{N}$  (also die unäre Kodierung von  $n$  und  $m$  durch  $\circ$  getrennt) das Prädikat  $n \leq m$  berechnet.

$$f(x) = \begin{cases} 1 \text{ (akzeptiert)} & n \leq m \\ 0 \text{ (nicht akzeptiert)} & \text{sonst} \end{cases}$$

Lösungsvorschlag



Der Automat auf flaci.com (FLACI: Formale Sprachen, abstrakte Automaten, Compiler und Interpreter) Ein Projekt der Hochschule Zittau/Görlitz und der Pädagogischen Hochschule Schwyz: [flaci.com/Ajf6mi7e7](http://flaci.com/Ajf6mi7e7)

## 66115 / 2021 / Frühjahr / Thema 2 / Aufgabe 2

### Ableitung einer Phrasenstrukturgrammatik

Die folgende Grammatik erzeugt die Sprache  $L = \{a^{2^n} \mid n \in \mathbb{N}\}$ , nicht kontextsensitiv ist:

$$G = (\{S, L, R, D\}, \{a\}, P, S)$$

$$P = \left\{ \right.$$

$$S \rightarrow LaR$$

$$L \rightarrow LD \mid \varepsilon$$

$$Da \rightarrow aaD$$

$$DR \rightarrow R$$

$$R \rightarrow \varepsilon$$

$$\left. \right\}$$

Leiten Sie das Wort  $a^8$  ab.

Lösungsvorschlag

$LaR \vdash LDaR \vdash LDDaR \vdash LDDDaR \vdash DDDaR \vdash DDaaDR \vdash DaaDaDR \vdash aaDaDaDR \vdash$   
 $aaaaDDaDR \vdash aaaaaaaaaDDDR \vdash aaaaaaaaaDDR \vdash aaaaaaaaaR \vdash aaaaaaa$

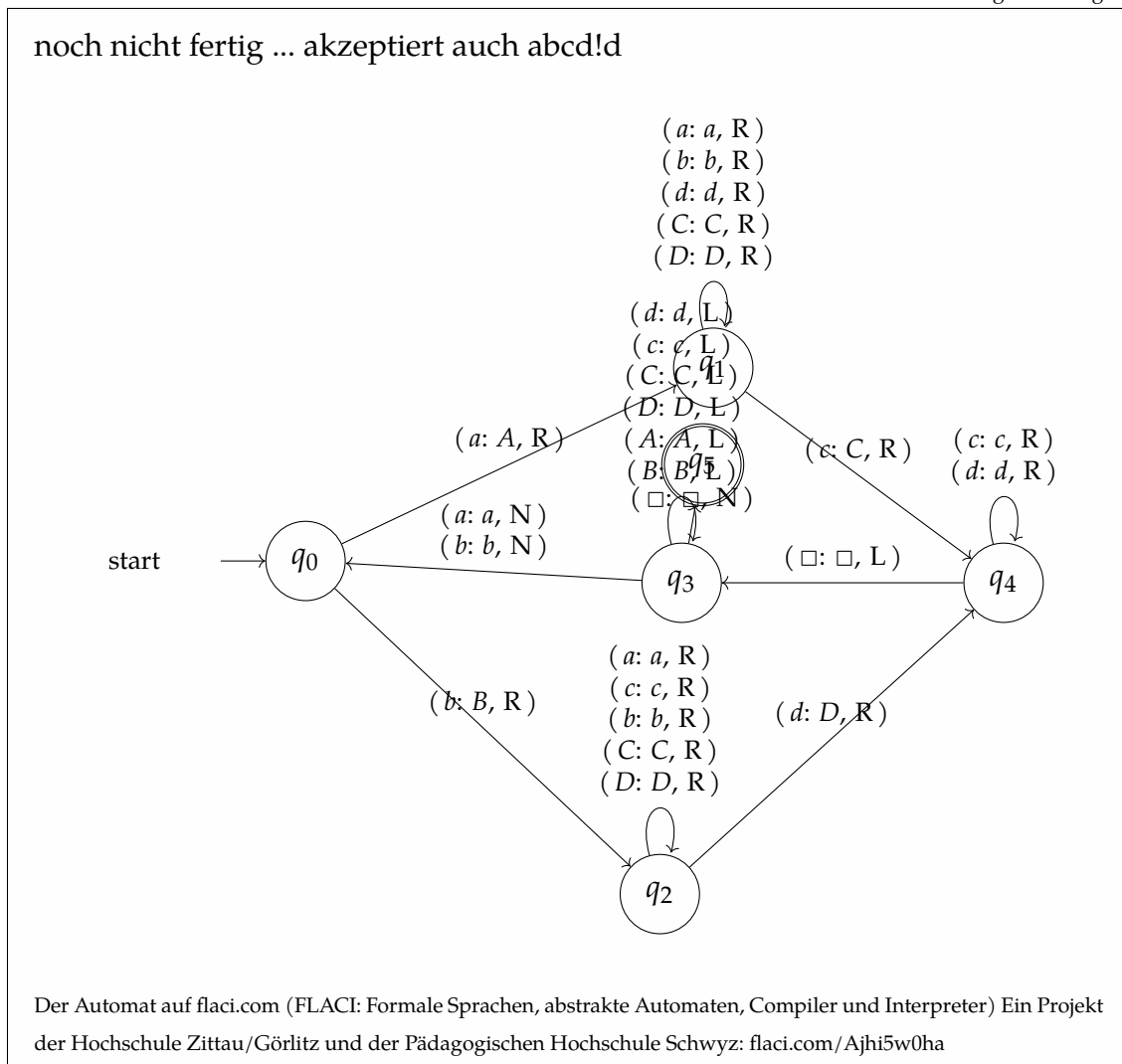
## 46115 / 2013 / Frühjahr / Thema 1 / Aufgabe 4

## Aufgabe 4

Sei  $L = \{uv \mid u \in \{a,b\}^*, v \in \{c,d\}^*, \#_a(u) = \#_c(v) \text{ und } \#_b(u) = \#_d(v)\}$  wobei  $\#_a(u)$  die Anzahl der in  $u$  vorkommenden  $a$ 's ist.

- (a) Geben Sie eine Turingmaschine  $M$  an, die  $L$  erkennt. Beschreiben Sie in Worten, wie Ihre Turingmaschine arbeitet.

Lösungsvorschlag



- (b) Welche Laufzeit (Zeitkomplexität) hat Ihre Turingmaschine (in O-Notation). Begründen Sie Ihre Angabe auf der Grundlage der Beschreibung.

## 66115 / 2010 / Herbst / Thema 2 / Aufgabe 7

Konstruieren Sie eine Turingmaschine  $M$  mit  $L(M) = L$ , wobei  $p, q \geq 1$ . Beschreiben Sie zusätzlich, wie  $M$  arbeitet (Stil:  $M$  liest das Zeichen  $a$  und speichert ....)

$$L = \{w \mid w \in \{a,b\}^*, w \text{ besteht aus } p \text{ Zeichen } a \text{ und aus } q \text{ Zeichen } b\}.$$

## 66115 / 2017 / Frühjahr / Thema 2 / Aufgabe 3

## (a) Primitiv rekursive Funktionen

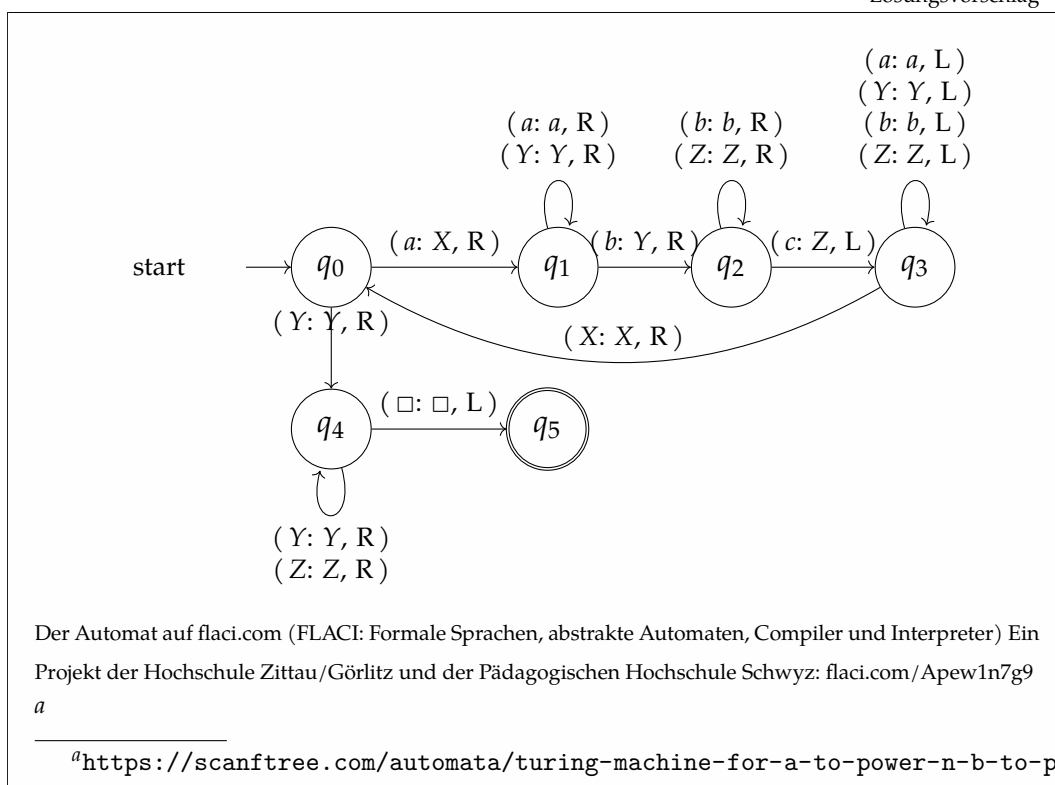
- (i) Zeigen Sie, dass die folgendermaßen definierte Funktion  $if: \mathbb{N} \times \mathbb{N} \times \mathbb{N} \rightarrow \mathbb{N}$  primitiv rekursiv ist.  
sonst
- (ii) Wir nehmen eine primitiv rekursive Funktion  $p: \mathbb{N} \rightarrow \mathbb{N}$  an und definieren  $g(n)$  als die Funktion, welche die größte Zahl  $i < n$  zurückliefert, für die  $p(i) = 0$  gilt. Falls kein solches  $i$  existiert, soll  $g(n) = 0$  gelten:  
 $a(n) = \max \{ i < n \mid p(i) = 0 \}$   
 $if(b, x, y) = ( \text{ falls } b=0$

Zeigen Sie, dass  $g: \mathbb{N} \rightarrow \mathbb{N}$  primitiv rekursiv ist. (Sie dürfen obige Funktion  $if$  als primitiv rekursiv voraussetzen.)

- (b) Sei  $\Sigma = \{a, b, c\}$  und  $L \subseteq \Sigma^*$  mit  $L = \{a^i b^j c^i \mid i \in \mathbb{N}\}$ .

- (i) Beschreiben Sie eine Turingmaschine, welche die Sprache  $L$  entscheidet. Eine textuelle Beschreibung der Konstruktionsidee ist ausreichend.

Lösungsvorschlag



- (ii) Geben Sie Zeit- und Speicherkomplexität (abhängig von der Länge der Eingabe) Ihrer Turingmaschine an.

**Speicherkomplexität**  $n$  (Das Eingabewort wird einmal überschrieben)

**Zeitkomplexität** the turing machine time complexity is the number of transition execution will executed is call time complexity of the turing machine. first we start we main loop execution is  $(n/3)-1$ . transition  $(a,x,R)$  from state 1 to 2 = 1. transition  $(a,a,R)$  and  $(y,y,R)$  on state 2 is =  $(n/3)-1$ . transition  $(b,y,R)$  from state 2 to 3 = 1. on state 3  $(b,b,R)$  and  $(z,z,R) = (n/3)-1$ . transition  $(c,z,L)$  from state 3 to 4 = 1. on state 4  $(y,y,L), (b,b,L), (z,z,L)$  and state 5  $(a,a,L) = (n/3)-1$ . transition  $(a,a,L)$  form state 4 to 5 = 1. transition  $(x,x,R)$  from 5 to 1 = 1 total  $(n+2)$  following transition will executed transition  $(a,x,R)$  from state 1 to 2 = 1. transition  $(y,y,R)$  on state 2 is =  $(n/3)-1$ . transition  $(b,y,R)$  from state 2 to 3 = 1. transition  $(z,z,R)$  on state 3 =  $(n/3)-1$  transition  $(c,z,L)$  from state 3 to 4 = 1. on state 4  $(y,y,L), (z,z,L)$  and state  $(n/3)-1$ . transition  $(x,x,R)$  from state 5 to 6 = 1 transition on state 6  $(y,y,R), (z,z,R) = (n/3)$  transition  $(d,d,R)$  from state 6 to 7 = 1 total =  $(4n/3)+2$  over alti time complexity  $(n+2)(n/3)-1 + (4n/3)+2$ <sup>a</sup>

<sup>a</sup>[https://www.youtube.com/watch?v=vwnz9e\\_Lrfo](https://www.youtube.com/watch?v=vwnz9e_Lrfo)

(c) Sei  $\Sigma = \{0,1\}$ . Jedes  $w \in \Sigma^*$  kodiert eine Turingmaschine  $M_w$ . Die von  $M_w$  berechnete Funktion bezeichnen wir mit  $\varphi_w(x)$ .

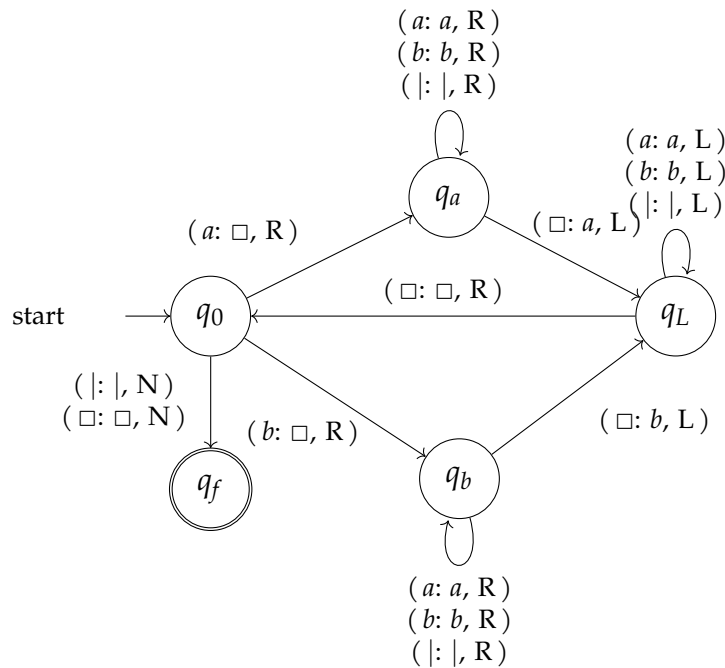
(i) Warum ist  $L = \{w \in \Sigma^* \mid \exists x: \varphi_w(x) = xx\}$  nicht entscheidbar?

(ii) Warum ist  $L = \{w \in \Sigma^* \mid \exists x: w = xx\}$  entscheidbar?

## 66115 / 2019 / Frühjahr / Thema 1 / Aufgabe 4

Wir betrachten die Turingmaschine  $M = (Z, \Sigma, \Gamma, \delta, z_0, \square, E)$ . Hierbei ist die Zustandsmenge  $Q = \{q_0, q_a, q_b, q_L, q_f\}$  mit Startzustand  $q_0$  und akzeptierenden Zuständen  $F = \{q_f\}$ . Das Eingabealphabet ist  $\Sigma = \{a, b, \}$ <sup>1</sup> das Bandalphabet ist  $\Gamma = \Sigma \cup \{\square\}$  mit Blank-Zeichen  $\square$  für leeres Feld. Die Übergangsfunktion  $\delta : Z \times \Gamma \rightarrow Z \times \Gamma \times \{L, R, N\}$ , wobei der Schreib-Lese-Kopf mit L nach links, mit N nicht und mit R nach rechts bewegt wird, ist durch folgende Tabelle gegeben (bspw. ist  $\delta(q_0, a) = (q_a, \square, R)$ ):

<sup>1</sup>In der Angabe ist das Trennzeichen ein „\$“. Wir verwenden stattdessen ein „|“, denn „\$“ ist eine Tex-Sonderzeichen und müsste deshalb ständig besonders behandelt werden.



Der Automat auf flaci.com (FLACI: Formale Sprachen, abstrakte Automaten, Compiler und Interpreter) Ein Projekt der Hochschule Zittau/Görlitz und der Pädagogischen Hochschule Schwyz: flaci.com/Aj54q4rd9

- (a) Die Notation  $(v, q, aw)$  beschreibt eine Konfiguration der Turingmaschine: der interne Zustand ist  $q$ , der Schreib-Lesekopf steht auf einem Feld mit  $a \in \Gamma$ , rechts vom Schreib-Lesekopf steht  $w \in \Gamma^*$ , links vom Schreib-Lesekopf steht  $v \in \Gamma^*$ .

Vervollständigen Sie die Folge von Konfigurationen, die die Turingmaschine bei Eingabe  $ab|$  bis zum Erreichen des Zustands  $q_f$  durchläuft. Sie können auch Ihre eigene Notation zur Darstellung von Konfigurationen verwenden.

$(\square, q_0, ab|) \vdash$   
 $(\square, q_a, \square b|) \vdash$   
 $(\square, q_a, b|) \vdash$   
 $\dots$

Lösungsvorschlag

$(\square, q_0, ab|) \vdash$   
 $(\square, q_a, \square b|) \vdash$   
 $(\square, q_a, b|) \vdash$   
 $(b, q_a, |) \vdash$   
 $(b|, q_L, a) \vdash$   
 $(b, q_L, |a) \vdash$   
 $(\square, q_L, \square b|a) \vdash$   
 $(\square, q_b, \square b|a) \vdash$   
 $(\square, q_b, \square|a) \vdash$   
 $(\square, q_b, |a) \vdash$   
 $(|, q_b, a) \vdash$   
 $(|a, q_L, b) \vdash$   
 $(|, q_L, ab) \vdash$



$$\begin{array}{l} (\square, q_L, |ab) \vdash \\ (\square, q_0, \square|ab) \vdash \\ (\square, q_f, |ab) \end{array}$$

Turing-Maschine

- (b) Sei  $w \in \{a, b\}^*$  beliebig. Mit welchem Bandinhalt terminiert die Turingmaschine bei Eingabe von  $w$ ? Geben Sie auch eine kurze Begründung an.

Lösungsvorschlag

Die Turingmaschine terminiert bei allen möglichen Wörtern  $w \in \{a, b\}^*$ , auch bei dem leeren Wort vor  $|$ . Die Turing-Maschine verschiebt alle  $a$ 's und  $b$ 's vor dem Trennzeichen  $|$  nach rechts. Ist das Trennzeichen  $|$  schließlich das erste Zeichen von links gesehen, dann terminiert die Maschine.

## 66115 / 2019 / Herbst / Thema 2 / Aufgabe 1

Gesucht ist eine Turing-Maschine mit genau einem beidseitig unendlichen Band, die die Funktion  $f : \mathbb{N} \rightarrow \mathbb{N}$  mit  $f(x) = 3x$  berechnet. Zu Beginn der Berechnung steht die Eingabe binär codiert auf dem Band, wobei der Kopf auf die linkeste Ziffer (most significant bit) zeigt. Am Ende der Berechnung soll der Funktionswert binär codiert auf dem Band stehen, wobei der Kopf auf ein beliebiges Feld zeigen darf.

- (a) Beschreiben Sie zunächst in Worten die Arbeitsweise Ihrer Maschine.

Lösungsvorschlag

$13 \cdot 3 = 0b1101 \cdot 0b11 = 39 = 0b100111$ :

$$\begin{array}{r} \phantom{0}1 \phantom{0}1 \phantom{0}0 \phantom{0}1 \\ \phantom{0}1 \phantom{0}1 \phantom{0}0 \phantom{0}1 \\ \hline \phantom{0}1 \phantom{0}0 \phantom{0}0 \phantom{0}1 \phantom{0}1 \phantom{0}1 \end{array}$$

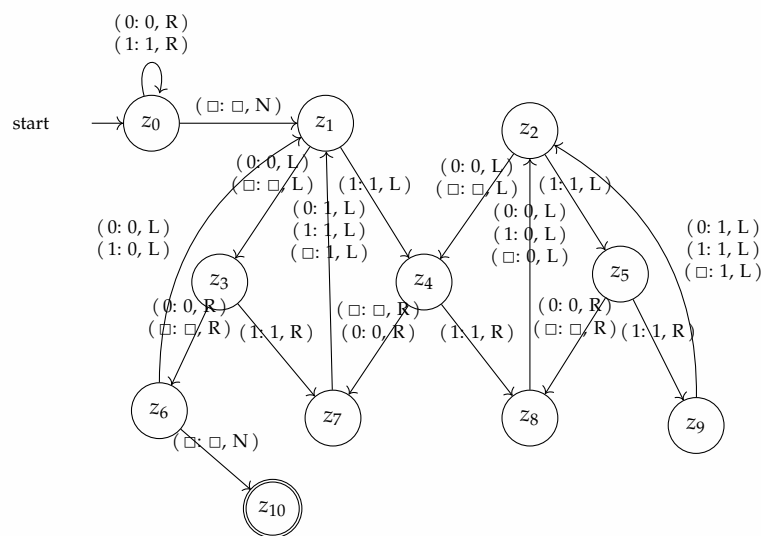
Die entworfene Turingmaschine imitiert die Vorgehensweise beim schriftlichen Multiplizieren. Die Maschine geht zunächst an das Leerzeichen am rechten Ende des Eingabewortes. Die Maschine bewegt sich nun zwei Schritte nach links und liest die Zahlen ein und addiert sie. Schließlich bewegt sich die Maschine einen Schritt nach rechts und schreibt das Ergebnis der Addition. Dabei wird das Eingabewort überschrieben allmählich überschrieben.

- (b) Geben Sie dann das kommentierte Programm der Turing-Maschine an und erklären Sie die Bedeutung der verwendeten Zustände.

Lösungsvorschlag

- $z_0$  An das rechte Ende des Eingabewortes gehen
- $z_1$  Übertrag 0
- $z_2$  Übertrag 1
- $z_3$  1. Additionsschritt: +0
- $z_4$  1. Additionsschritt: +1
- $z_5$  1. Additionsschritt: +2
- $z_6$  2. Additionsschritt: +0
- $z_7$  2. Additionsschritt: +1
- $z_8$  2. Additionsschritt: +2
- $z_9$  2. Additionsschritt: +3
- $z_{10}$  Endzustand

Nicht sehr übersichtlich hier: Im Anhang findet sich die JSON-Datei für flaci.com



Der Automat auf flaci.com (FLACI: Formale Sprachen, abstrakte Automaten, Compiler und Interpreter) Ein Projekt der Hochschule Zittau/Görlitz und der Pädagogischen Hochschule Schwyz: [flaci.com/Ahjkw50kg](http://flaci.com/Ahjkw50kg)

# Berechenbarkeit

66115 / 2019 / Herbst / Thema 2 / Aufgabe 1

## GOTO-Programme

- (a) Terminieren GOTO-Programme immer? Begründe Deine Antwort.

Lösungsvorschlag

GOTO-Programme terminieren nicht immer.

**Variante 1:** Die Menge der GOTO-Programme ist identisch mit der Menge der WHILE-Programme. Da WHILE-Programme partielle Funktionen beschreiben und diese nicht für alle Eingaben terminieren, terminieren GOTO-Programme ebenfalls nicht für alle Eingaben.

**Variante 2:** Die charakteristische Funktion einer semi-entscheidbaren Sprache ist Turing- bzw. GOTO-berechenbar, d.h. zu jeder semi-entscheidbaren Sprache gibt es eine Turing-Maschine. GOTO-Programme können Turing-Maschinen simulieren. Da hier von einer nur semi-entscheidbaren Sprache ausgegangen wird, terminiert das GOTO-Programm nicht, falls die Eingabe  $x$  kein Element der Sprache ist.

- (b) Gebe ein GOTO-Programm an, dass die Summe dreier Zahlen berechnen.

Lösungsvorschlag

```
Eingabe x_1, x_2, x_3;
x_0 := x_1;
IF x_2 = 0 GOTO Z6;
  x_0 := x_0 + 1;
  x_2 := x_2 - 1;
GOTO Z2;
IF x_3 = 0 GOTO Z10;
  x_0 := x_0 + 1;
  x_3 := x_3 - 1;
GOTO Z6;
END;
Ausgabe: x_0
```

- (c) Gegeben ist das GOTO-Programm:

```
x_4 := x_1;
IF x_4 = 0 GOTO Z10;
  x_5 := x_2;
IF x_5 = 0 GOTO Z8;
  x_3 := x_3 + 1;
  x_5 := x_5 - 1;
```

```
GOTO Z4;
  x_4 := x_4 - 1;
GOTO Z2;
  x_5 := x_5 - 1
```

(i) Was berechnet das Programm?

Lösungsvorschlag

$$f(n, m) = n * m$$

(ii) Übertrage das Programm in ein WHILE-Programm.

Lösungsvorschlag

```
Eingabe x_1, x_2 :
x_4 := x_1;
WHILE x_4 <> 0 DO
  x_5 := x_2;
  WHILE x_5 <> 0 DO
    x_3 := x_3 + 1;
    x_5 := x_5 - 1
  END;
  x_4 := x_4 - 1
END
Ausgabe x_0

Eingabe : x_1, x_2
x_0 := mult ( x_1, x_2 );
Ausgabe : x_0
```

## 66115 / 2019 / Herbst / Thema 2 / Aufgabe 1

(a) Geben Sie ein LOOP-Programm an, das die Funktion  $f(n) = n!$  berechnet.

Lösungsvorschlag

```
x_2 := 1;
LOOP x1 DO
  x_3 := x_3 + 1;
  x_2 := x_2 * x_3;
END
x_3 := 0;
RETURN x_2;
```

(b) Beweisen Sie:

Ist  $f : N \rightarrow N$  LOOP-berechenbar, so ist auch  $g : N \rightarrow N$  mit  $g(n) = f(i)$  LOOP-berechenbar.

Bei einem LOOP-Programm der Form  $\text{LOOP } x_i \text{ DO } P \text{ END}$  wird das Programm  $P$  so oft ausgeführt, wie der Wert der Variablen  $x_i$  zu Beginn angibt. Beweis:

```
x_0 := 0;
i := 0;
LOOP n DO
  i := i + 1;
  y := f(i);
  x_0 := x_0 + y;
END
RETURN x_0;
```

ist LOOP-berechenbar, da  $f(n)$  LOOP-berechenbar ist.

## 66115 / 2019 / Herbst / Thema 2 / Aufgabe 1

### Vorlesungsaufgaben

### LOOP-Implementierung

(a) Geben Sie eine LOOP-Implementierung für

(i)  $\text{add}(x_i, x_j)$

Lösungsvorschlag

```
x_0 := x_i;
LOOP x_j DO
  x_0 := succ(x_0);
END
```

(ii)  $\text{mult}(x_i, x_j)$

Lösungsvorschlag

```
x_0 := x_i;
LOOP x_j DO
  x_0 := add(x_0, x_i);
END
```

(iii)  $\text{power}(x_i, x_j)$

Lösungsvorschlag

```
x_0 := succ(0);
LOOP x_j DO
  x_0 := mult(x_0, x_i);
END
```

(iv)  $\text{hyper}(x_i, x_j)$

Lösungsvorschlag

Berechenbarkeit

```

x_0 := succ(0);
LOOP x_j DO
  x_0 := power(x_i, x_0);
END

```

(v)  $2^{x_i}$ 

Lösungsvorschlag

Mit *power*

```

x_0 := power(2, x_i);

```

Mit *mult*

```

x_0 := 1;
x_2 := 2;
LOOP x_i DO
  x_0 := mult(x_0, x_2);
END

```

an.

- (b) Beweisen Sie, dass der größte gemeinsame Teiler zweier natürlicher Zahlen LOOP-berechenbar ist.

Lösungsvorschlag

```

ggT(x_1, x_2)

x_3 := MAX(x_1, x_2);
x_4 := MIN(x_1, x_2);

LOOP x_4 DO
  x_5 := x_3 - x_4;
  x_3 := MAX(x_4, x_5);
  x_4 := MIN(x_4, x_5);
END
x_0 := x_3;

```

## 66115 / 2019 / Herbst / Thema 2 / Aufgabe 1

### Primitiv-rekursiv

Lösungsvorschlag

Begründe, dass folgende Funktionen primitiv-rekursiv sind: 1.  $f(x) = x!$  (1 wenn  $x > 0$ ) 2.  $\text{sig}(x) = 0$  sonst

Lösungsvorschlag

Begründung durch eine Angabe einer Funktion: 1.  $f(0) = 1, f(n+1) = \text{mult}(n+1, f(n))$  2.  $\text{sig}(0) = 0, \text{sig}(n+1) = 1$

Gebe eine konkrete primitiv-rekursive Implementierung für  $\text{if } x_1 \text{ then } x_2 \text{ else } x_3$  an. Wobei wie bei Programmiersprachen  $x_1$  als wahr gilt, wenn der Wert nicht Null ist.

Lösungsvorschlag

Zusätzlich werden die folgenden Funktionen festgelegt:

$\text{isZero}(0) = 1$   $\text{isZero}(n) = \text{isZero}(n + 1) = 0$   $\text{not}(n) = 1 - n$   $\text{ite}(x_1, x_2, x_3)$   
 $= \text{isZero}(x_1) * x_3 + \text{not}(\text{isZero}(x_1)) * x_2$

## 66115 / 2019 / Herbst / Thema 2 / Aufgabe 1

### Vorlesungsaufgaben

### Turing-berechenbar

- (a) Zeige, dass es nur abzählbar viele Turingmaschinen gibt.
- (b) Turing-berechenbar
  - (i) Definiere eine berechenbare Funktion  $f : N \rightarrow N$  mit entscheidbarem
  - (ii) Definitionsbereich und unentscheidbarem Wertebereich. Untersuche folgende Aussagen
    - i. Jede berechenbare Funktion  $h : N \rightarrow N$  mit endlichem Wertebereich besitzt einen entscheidbaren Definitionsbereich.
    - ii. Jede berechenbare Funktion  $g : N \rightarrow N$  mit endlichem Definitionsbereich besitzt einen entscheidbaren Wertebereich.

## 66115 / 2019 / Herbst / Thema 2 / Aufgabe 1

### TURING-berechenbar

Lösungsvorschlag

Ja, denn folgende TM führt die Berechnung aus. T liest eine links vom Trennzeichen stehende 1, ersetzt sie durch eine Null und fährt im Zustand Z 1 so lange nach rechts bis eine 0 erscheint. Diese wird gelöscht und dann im Zustand Z 2 nach links gewandert, um die dort am Anfang der Einserkette stehende 0 zu löschen. Nach Abarbeiten der  $n$  Einsen dürfte dann rechts des Trennzeichens nur noch eine 1 stehen. Dies wird nun mit Hilfe der restlichen Zustände überprüft. Steht nun noch eine 1 auf dem Band (also rechts daneben #), so macht T einen Schritt nach links und bleibt unter der 1 stehen. Findet T noch eine NULL, so bleibt sie bei dieser stehen.<sup>a</sup>

<sup>a</sup><https://flaci.com/Aputs940c>

## 66115 / 2019 / Herbst / Thema 2 / Aufgabe 1

## Vorlesungsaufgaben WHILE-Programm

Geben Sie ein WHILE-Programm an, dass

-  $2^{x_i}$

Lösungsvorschlag

Ausnutzen der 2er-Potenzeigenschaft:

$$2^1 = 1 + 1 = 2$$

$$2^2 = 2 + 2 = 4$$

$$2^3 = 4 + 4 = 8$$

Hier werden nur die elementaren Bestandteile der WHILE-Sprache ausgenutzt.

```
x_2 := 1;
WHILE x_1 DO
  x_3 := x_2;
  WHILE x_3 DO
    x_2 := x_2 + 1;
    x_3 := x_3 - 1;
  END
  x_1 := x_1 - 1;
END
x_0 := x_2;
```

-  $\text{ggT}(x_i, x_j)$

Lösungsvorschlag

Zusätzliche Voraussetzungen:

```
x_1 > x_2;
MOD(x_1, x_2);

x_0 := MOD(x_1, x_2);
WHILE x_0 DO
  x_1 := x_2;
  x_2 := x_0;
  x_0 := MOD(x_1, x_2);
END
```

berechnet.

## 66115 / 2019 / Herbst / Thema 2 / Aufgabe 1

Bestimme jeweils, ob die angegebene Funktion WHILE-berechenbar ist:

(a)  $x \rightarrow 2^x$



```

erg = 1;
WHILE x != 0 DO
  erg = erg * 2;
  x = x - 1;
END;
return erg;

```

(b)  $\text{ggT}(n, m)$ ,

also der größte gemeinsame Teiler. Sie dürfen die (ganzzahligen) Operationen  $+$ ,  $*$  und  $/$  verwenden, wobei das Minus, wie üblich, eingeschränkt ist.

Es bietet sich an, zunächst die modulo Operation

$$x_i := x_j \% x_k$$

durch folgendes WHILE-Programm zu definieren:

```

x_{n+1} := x_j / x_k;
x_{n+2} := x_{n+1} * x_k;
x_i := x_j - x_{n+2};

```

Wobei  $x_{n+1}$  und  $x_{n+2}$  im Rest des Programmes nicht verwendet werden sollen. Mit der Modulo Operation kann man nun z. B. einfach den euklidischen Algorithmus verwenden (Eingabe seien  $x_1$  und  $x_2$ , Ausgabe ist  $x_1$ ):

```

WHILE x_2 != 0 DO
  x_3 := x_1 \% x_2;
  x_1 := x_2 + 0;
  x_2 := x_3 + 0;
END

```

(c)  $\text{if } x_i \neq 0 \text{ then } P_1 \text{ else } P_2 \text{ fi}$ 

mit der üblichen Semantik. Als Nachweis kann jeweils ein WHILE-Programm angegeben werden.

Sei  $x_n$  die höchste in  $P_1$  bzw.  $P_2$  vorkommende Variable (o. E.  $i \leq n$ ).

```

x_{n+1} := x_i + 0;
x_{n+2} := 1;
WHILE x_{n+1} != 0 DO
  x_{n+1} := 0;
  x_{n+2} := 0;
  P_1;
END
WHILE x_{n+2} != 0 DO
  x_{n+2} := 0;
  P_2;

```

**66112 / 2002 / Frühjahr / Thema 2 / Aufgabe 5**

Zeige, dass die Präfixrelation ( $\text{pr\"{a}fix}(u,v) : \leftrightarrow \exists w \in \{a, b\}^* : uw = v$ ) auf  $\{a, b\}^*$  entscheidbar ist.

**66112 / 2004 / Herbst / Thema 2 / Aufgabe 3**

entscheidbar?

**66115 / 2015 / Frühjahr / Thema 2 / Aufgabe 4**

Sei  $M_0, M_1, \dots$  eine Gödelisierung aller Registermaschinen (RAMs). Geben Sie für die folgenden Mengen  $D_1, D_2, D_3$  an, ob sie entscheidbar oder aufzählbar sind. Begründen Sie Ihre Behauptungen, wobei Sie die Aufzählbarkeit und Unentscheidbarkeit des speziellen Halteproblems  $K_0 = \{x \in \mathbb{N} \mid M_x \text{ h\"{a}lt bei Eingabe } x\}$  verwenden dürfen.  $D_1 = \{x \in \mathbb{N} \mid x < 9973 \text{ und } M_x \text{ h\"{a}lt bei Eingabe } x\}$   $D_2 = \{x \in \mathbb{N} \mid x \geq 9973 \text{ und } M_x \text{ h\"{a}lt bei Eingabe } x\}$   $D_3 = \{x \in \mathbb{N} \mid M_x \text{ h\"{a}lt nicht bei Eingabe } x\}$

Lösungsvorschlag

$D_1$  ist eine endliche Menge und damit entscheidbar. Auch eine endliche Teilmenge des Halteproblems. Anschaulich kann man sich dies so vorstellen: Man stellt dem Rechner eine Liste zur Verfügung, die alle haltenden Maschinen  $M_x$  mit  $x < 9973$  enthält. Diese Liste kann zum Beispiel vorab von einem Menschen erstellt worden sein, denn die Menge der zu prüfenden Programme ist endlich.

$D_2 = \{x \geq 9973 \mid M_x \text{ h\"{a}lt}\}$  entscheidbar,  $L_{\text{halt}}$  semi-entscheidbar  $\rightarrow$  semi-entscheidbar (Hier wäre auch eine Argumentation über die Cantorsche Paarungsfunktion möglich). Es ist weiterhin nicht entscheidbar. Dazu betrachten wir die Reduktion des speziellen Halteproblems  $H_0 : H_0 \leq D_2$  Für alle  $x < 9973$  lassen wir  $M_x$  durch eine Turingmaschine  $M_y$  simulieren, die eine höhere Nummer hat.

$D_3$  ist unentscheidbar, denn angenommen  $D_3$  wäre semi-entscheidbar, dann würde sofort folgen, dass  $L_{\text{halt}}$  entscheidbar ist, da aus der Semientscheidbarkeit von  $L_{\text{halt}}$  und  $L_{\text{halt}}$  die Entscheidbarkeit von  $L_{\text{halt}}$  folgen würde

**66115 / 2016 / Frühjahr / Thema 2 / Aufgabe 2**

Beantworten Sie kurz, präzise und mit Begründung folgende Fragen: (Die Begründungen müssen keine formellen mathematischen Beweise sein)

- (a) Warum genügt es, sich auf Funktionen zu beschränken, die natürliche Zahlen auf natürliche Zahlen abbilden, wenn man untersuchen will, was heutige Computer im Prinzip berechnen können?

Lösungsvorschlag

Jede WHILE-berechenbare Funktion ist turing-berechenbar. Jede turing-berechenbare Funktion ist WHILE-berechenbar. Jede nichtdeterministische Turing-Maschine kann durch eine deterministische Turing-Maschine simuliert werden.

- (b) Was besagt die Church-Turing-These? Könnte man sie beweisen oder widerlegen?

Lösungsvorschlag

Die Church-Turing-These besagt, dass die Klasse der turing-berechenbaren Funktionen mit der Klasse der intuitiv berechenbaren Funktionen übereinstimmt. Die These kann nicht bewiesen oder widerlegt werden, weil es sich bei dem Begriff „*intuitiv berechenbare Funktion*“ um keinen mathematisch exakt definierten Begriff handelt. Würde man ihn genau definierten, würde ein konkretes Berechnungsmodell festgelegt werden, was der Kernaussage dieses Begriffes widersprechen würde.

- (c) Für reelle Zahlen, wie z. B.  $\pi$ , lässt sich die Dezimaldarstellung durch entsprechende Programme beliebig genau approximieren. Gilt das für alle reellen Zahlen, lässt sich für jede reelle Zahl die Dezimaldarstellung mit entsprechenden Programmen beliebig genau approximieren?

Lösungsvorschlag

Ja mit einer Berechnungsvorschrift, ja solange der Speicherplatz reicht, z. B. mittels Intervallschachtelung.

- (d) Was ist für die Berechnungskraft der wesentliche Unterschied zwischen While-Berechenbarkeit und Loop-Berechenbarkeit.

Lösungsvorschlag

Alle LOOP-Programme sind mathematisch betrachtet totale Funktionen, die terminieren immer. WHILE-Programme hingegen partiellen Funktionen, die nicht für alle Eingabekombinationen terminieren.

- (e) Die Ackermannfunktion ist ein Beispiel einer totalen Funktion, die While-berechenbar, aber nicht Loop-berechenbar ist. Sie verallgemeinert die Idee, dass Multiplikation die wiederholte Addition ist, Exponentiation die wiederholte Multiplikation, Hyperexponentiation die wiederholte Exponentiation usw. Die Stufe dieser hyperhyper ... Exponentiation ist ein Parameter der Ackermannfunktion. Generieren Sie aus dieser Idee ein Argument, das illustriert, warum die Ackermannfunktion nicht Loop-berechenbar ist.

Lösungsvorschlag

Jedes LOOP-Programm benötigt zur Berechnung der Funktion  $x \uparrow^n y$  mindestens  $n + 2$  Schleifen.

Gäbe es ein LOOP-Programm, das  $ack(n, m)$  tatsächlich für beliebige Werte von  $n$  berechnet, so müsste dieses — im Widerspruch zum endlichen Aufbau eines Loop-Programms — unendlich viele Schleifenkonstrukte enthalten.

**Hyperexponentiation**

- $x \uparrow^{-1} y = x + y$
- $x \uparrow^0 y = x \cdot y$
- $x \uparrow^1 y = x^y$
- $x \uparrow^2 y = x^{y^y}$

- (f) Geben Sie ein Beispiel einer Menge an, die abzählbar, aber nicht rekursiv aufzählbar ist, und begründen Sie es.

Lösungsvorschlag

- Die Menge der Primzahlen. Die Primzahl sind unendlich groß und könnten abgezählt werden. Bei Primzahlen gibt es keine Berechnungsvorschrift, die z. b. die 7. Primzahl berechnet.
- Die Menge der Turningmaschine
- Diagonalsprache
- Das Komplement des Halteproblems. Die dem Halteproblem zugrundeliegende Lösungsmenge ist rekursiv aufzählbar. Wäre das Komplement des Halteproblems auch rekursiv aufzählbar, dann wäre das Halteproblem entscheidbar, was es aber nicht ist.<sup>a</sup>

<sup>a</sup><https://www.youtube.com/watch?v=om4ZT0eQuD0>

- (g) Wie ist der Zusammenhang zwischen rekursiv aufzählbar und semi-entscheidbar?

Lösungsvorschlag

Die beiden Begriffe sind äquivalent.<sup>a</sup>

<sup>a</sup><https://www.youtube.com/watch?v=om4ZT0eQuD0>

**66115 / 2016 / Herbst / Thema 2 / Aufgabe 3**

Sei  $M_0, M_1, \dots$  eine Registermaschinen (RAMs). Beantworten Sie folgende Fragen zur Aufzählbarkeit und Entscheidbarkeit. Beweisen Sie Ihre Antwort.

**Exkurs: Registermaschinen (RAMs)**

Die Random Access Machine (kurz RAM) ist eine spezielle Art von Registermaschine. Sie hat die Fähigkeit der indirekten Adressierung der Register.

Die Random Access Machine besteht aus:

- einem Programm bestehend aus endlich vielen durchnummerierten Befehlen (beginnend mit Nummer 1)
- einem Befehlszähler  $b$
- einem Akkumulator  $c(0)$
- und einem unendlich großen Speicher aus durchnummerierten Speicherzellen (Registern)  $c(1), c(2), c(3), \dots$

Jedes Register (einschließlich  $b$  und  $c(0)$ ) speichert eine beliebig große natürliche Zahl.

(a) Ist folgende Menge entscheidbar?

$$A = \{x \in \mathbb{N} \mid x = 100 \text{ oder } M_x \text{ hält bei Eingabe } x\}$$

Lösungsvorschlag

Ja,  $x \geq 100$  ist entscheidbar und aufgrund des „oder“ ist die 2. Bedingung nur für  $x < 100$  relevant. Da  $x < 100$  eine endliche Menge darstellt, kann eine endliche Liste geführt werden und ein Experte kann für jeden Fall entscheiden, ob  $M_x$  hält oder nicht, somit ist  $A$  entscheidbar.

(b) Ist folgende Menge entscheidbar?

$$B = \{(x, y) \in \mathbb{N} \times \mathbb{N} \mid M_x \text{ hält bei Eingabe } x \text{ genau dann, wenn } M_y \text{ bei Eingabe } y \text{ hält}\}$$

Lösungsvorschlag

Nein. Dieses Problem entspricht der parallelen Ausführung des Halteproblems auf zwei Bändern. Das Halteproblem ist unentscheidbar, damit ist auch die parallele Ausführung des Halteproblems und damit  $B$  unentscheidbar.

(c) Ist folgende Menge aufzählbar?

$$C = \{x \in \mathbb{N} \mid M_x \text{ hält bei Eingabe } 0 \text{ mit dem Ergebnis } 1\}$$

Lösungsvorschlag

Ja, die Menge ist aufzählbar, da die Menge aller Turingmaschinen aufzählbar und über natürliche Zahlen definiert ist (die wiederum aufzählbar sind).

## 66115 / 2020 / Frühjahr / Thema 2 / Aufgabe 4

$A = \{(M) \mid M \text{ ist Turingmaschine, die bei Eingabe } 101 \text{ hält}\}$ . Dabei bezeichnet  $(M)$  die Gödelnummer der Turingmaschine  $M$ .

(a) Zeigen Sie, dass  $A$  unentscheidbar ist.

Lösungsvorschlag

Reduktionsbeweis von  $H_0 \leq A$ : TM  $U$

- (i) Die zu  $\square' \in \square^0$  passende TM  $\square * \in \square^*$  aus  $A$  suchen mit  $\langle \square' \rangle = \langle \square * \rangle$
- (ii) 101 auf das Band schreiben
- (iii)  $\square *$  auf 101 starten

Damit könnte  $U$   $H_0$  entscheiden, was aber ein Widerspruch zu  $H_0$  semi-entscheidbar ist. Damit ist  $A$  ebenfalls semi-entscheidbar.

(b) Zeigen Sie, dass  $A$  semi-entscheidbar ist.

Lösungsvorschlag

siehe a)

- (c) Ist das Komplement  $A^c$  von  $A$  entscheidbar? Ist es semi-entscheidbar? Begründen Sie Ihre Antworten.

Hinweis: Sie können die Aussagen aus Teilaufgabe a) und b) verwenden, auch wenn Sie sie nicht bewiesen haben.

Lösungsvorschlag

Wenn  $A$  unentscheidbar ist, dann kann entweder  $A$  oder  $A^c$  semi-entscheidbar sein. Wären beide semi-entscheidbar, dann wäre  $A$  aber ebenfalls entscheidbar, was aber nach Voraussetzung ausgeschlossen ist.

# Entscheidbarkeit

## Post'sches Korrespondenzproblem (PCP) und Modifiziertes Post'sches Korrespondenzproblem (MPCP)

Post'sches Korrespondenzproblem

PCP
<b>Gegeben:</b> $(x_1, y_1), \dots, (x_n, y_n)$ mit $x_1, y_1 \in \Sigma^+$
<b>Frage:</b> Gibt es eine Folge $i_1, i_2, \dots, i_k \in \mathbb{N}$ mit der Eigenschaft $x_{i_1}x_{i_2} \dots x_{i_k} = y_{i_1}y_{i_2} \dots y_{i_k}$

Modifiziertes Post'sches Korrespondenzproblem

MPCP
<b>Gegeben:</b> $(x_1, y_1), \dots, (x_n, y_n)$ mit $x_1, y_1 \in \Sigma^+$
<b>Frage:</b> Gibt es eine Folge $i_1, i_2, \dots, i_k \in \mathbb{N}$ mit der Eigenschaft $x_{i_1}x_{i_2} \dots x_{i_k} = y_{i_1}y_{i_2} \dots y_{i_k}$

### 46115 / 2021 / Frühjahr / Thema 1 / Teilaufgabe 1 / Aufgabe 4

Beweisen Sie, dass das folgende Entscheidungsproblem semi-entscheidbar ist.

Eingabe: Eine Turingmaschine  $M$  Aufgabe: Entscheiden Sie, ob ein Eingabewort existiert, auf das  $M$  hält.

### 66115 / 2013 / Herbst / Thema 1 / Aufgabe 3

Wir betrachten das wie folgt definierte Problem DOPP:

**GEGEBEN:** Eine deterministische Turingmaschine  $M$ , eine Eingabe  $x$  (für  $M$ ), ein Zustand  $q$  (von  $M$ ).

**GEFRAGT:** Wird der Zustand  $q$  bei der Berechnung von  $M$  auf  $x$  mindestens zweimal besucht?

- (a) Zeigen Sie durch Angabe einer Reduktion vom Halteproblem, dass DOPP unentscheidbar.
- (b) Begründen Sie, dass DOPP rekursiv aufzählbar (semi-entscheidbar) ist.

Die Reduktion  $f: \text{HALTE} \leq_p \text{DOPP}$ , bildet  $c(M), w$  auf  $c(M'), xw, q$  ab, wobei  $M'$  eine Turingmaschine mit folgendem Verhalten ist:

- Sie verfügt über den neuen Startzustand  $q$
- Sie erweitert das Wort  $w$  vorne um ein Zeichen  $x \notin \Gamma_M$  (dies ist nicht unbedingt notwendig, aber schöner, damit die neue Maschine auch noch terminiert)
- Für  $q$  wird die Regel  $(q, x) \rightarrow (z_0, \square, R)$ , wobei  $z_0$  der Startzustand von  $M$  ist
- Alle Endzustände  $z$  von  $M$  erhalten eine neue Regel  $(z, \square) \rightarrow (q, \square, N)$

Hierbei wird davon ausgegangen, dass das Halteproblem eine Turingmaschine mit Endzuständen als Eingabe hat und Aufgrund der Akzeptanz eines Wortes in diesem Fall das Zeichen gelöscht wird.

## 66115 / 2014 / Frühjahr / Thema 1 / Aufgabe 5

- (a) Definieren Sie die zum Halteproblem für Turing-Maschinen bei fester Eingabe  $m \in \mathbb{N}_0 = \{0, 1, 2, \dots\}$  gehörende Menge  $H_m$ .

Lösungsvorschlag

$H_m = \{c(M) \in \mathbb{N} \mid c(M) \text{ hält auf Eingabe } m\}$ , wobei  $c(M)$  der Codierung der Turingmaschine (Gödelnummer) entspricht.

- (b) Gegeben sei das folgende Problem  $E$ :

Entscheiden Sie, ob es für die deterministische Turing-Maschine mit der Gödelnummer  $n$  eine Eingabe  $w \in \mathbb{N}_0$  gibt, so dass  $w$  eine gerade Zahl ist und die Maschine  $n$  gestartet mit  $w$  hält.

Zeigen Sie, dass  $E$  nicht entscheidbar ist. Benutzen Sie, dass  $H_m$  aus (a) für jedes  $m \in \mathbb{N}_0$  nicht entscheidbar ist.

Lösungsvorschlag

Wir zeigen dies durch Reduktion  $H_2 \leq E$ :

- Berechenbare Funktion  $f$ : lösche Eingabe, schreibe eine 2 und starte dich selbst.
- $M$  ist eine Turingmaschine, die  $E$  entscheidet.
- $x \in H_2$  (Quellcode der Programme, die auf die Eingabe von 2 halten)
- $M_x$  (kompiliertes Programm, TM)
- Für alle  $x \in H_2$  gilt,  $M_x$  hält auf Eingabe von 2  $\Leftrightarrow f(x) = c(M) \in E$ . Denn sofern die ursprüngliche Maschine auf das Wort 2 hält, hält  $M$  auf alle Eingaben und somit auch auf Eingaben gerader Zahlen. Hält die ursprüngliche Maschine  $M$  nicht auf die Eingabe der Zahl 2, so hält  $M$  auf keine Eingabe.

- (c) Zeigen Sie, dass das Problem  $E$  aus (b) partiell-entscheidbar (= rekursiv aufzählbar) ist.



**66115 / 2017 / Frühjahr / Thema 1 / Aufgabe 6**

Es sei  $E$  die Menge aller (geeignet codierten) Turingmaschinen  $M$  mit folgender Eigenschaft: Es gibt eine Eingabe  $w$ , so dass  $M$  gestartet auf  $w$  mindestens 1000 Schritte rechnet und dann irgendwann hält.

Das Halteproblem auf leerer Eingabe  $H_0$  ist definiert als die Menge aller Turingmaschinen, die auf leerer Eingabe gestartet, irgendwann halten.

- (a) Zeigen Sie, dass  $E$  unentscheidbar ist (etwa durch Reduktion vom Halteproblem  $H_0$ ).

Lösungsvorschlag

zu zeigen:  $L_H \leq L \rightarrow L$  ist genauso unentscheidbar wie  $L_H$

Eingabeinstanzen von  $L_H(TM(M), u)$  durch Funktion umbauen in Eingabeinstanzen von  $L(TM(M'))$ .

Idee: Turingmaschine so modifizieren, dass sie zunächst 1000 Schritte macht und dann  $M$  auf  $u$  startet.

Dazu definieren wir die Funktion  $f : \Sigma^* \rightarrow \Sigma^*$  wie folgt:

$$f(u) = \begin{cases} c(M') & \text{falls } u = c(M')w \text{ ist für eine Turingmaschine } M \text{ und Eingabe } w \\ 0 & \text{sonst} \end{cases}$$

Dabei sei  $M'$  eine Turingmaschine, die sich wie folgt verhält:

- (i) Geht 1000 Schritte nach rechts
- (ii) Schreibt festes Wort  $w$  (für  $M'$  ist  $w$  demnach fest!)
- (iii) Startet  $M$

**total:** ja

**berechenbar:** Syntaxcheck, 1000 Schritte über 1000 weitere Zustände realisierbar

**Korrektheit:**  $u \in L_{\text{halt}} \Leftrightarrow u = c(M)w$  für TM  $M$ , die auf  $w$  hält  $\Leftrightarrow f(u) = c(M')$ , wobei  $M'$  1000 Schritte macht und dann hält  $\Leftrightarrow f(u) \in L$

- (b) Begründen Sie, dass  $E$  partiell entscheidbar ist.
- (c) Geben Sie ein Problem an, welches nicht einmal partiell entscheidbar ist.

**66115 / 2020 / Herbst / Thema 1 / Teilaufgabe 1 / Aufgabe 4**

Geben Sie für jede der folgenden Mengen an, ob sie entscheidbar ist oder nicht. Dabei ist  $\sigma_w$ , die Funktion, die von der Turingmaschine berechnet wird, die durch das Wort  $w$  kodiert wird. Beweisen Sie Ihre Behauptungen.

$$(a) L_1 = \{ w \in \Sigma^* \mid \sigma_w(0) = 0 \}$$

Lösungsvorschlag

Nicht entscheidbar wegen dem Halteproblem.

$$(b) L_2 = \{ w \in \Sigma^* \mid \sigma_w(w) = w \}$$

Lösungsvorschlag

Nicht entscheidbar wegen dem Halteproblem.

$$(c) L_3 = \{ w \in \Sigma^* \mid \sigma_0(0) = w \}$$

Lösungsvorschlag

Entscheidbar wegen  $\sigma_0$ .

### 66115 / 2021 / Frühjahr / Thema 2 / Teilaufgabe 1 / Aufgabe 3

Wir betrachten eine Gödelisierung von Turingmaschinen und bezeichnen mit  $M_w$  die Turingmaschine, die gemäß der Kodierung des Binärworts  $w$  kodiert wird. Außerdem bezeichnen wir mit  $M_w(x)$  die Ausgabe der Maschine  $M_w$  bei Eingabe  $x$ . Sie dürfen davon ausgehen, dass  $x$  immer ein Binärstring ist. Der bekannte Satz von Rice sagt:

Sei  $S$  eine Menge berechenbarer Funktionen mit  $\emptyset \neq S \neq \mathcal{R}$ , wobei  $\mathcal{R}$  die Menge aller berechenbaren Funktionen ist. Dann ist die Sprache  $L = \{ w \mid f_{M_w} \in S \}$  unentscheidbar.

Hier ist  $f_{M_w}$  die von  $M_w$  berechnete Funktion.

Zeigen Sie für jede der nachfolgenden Sprachen über dem Alphabet  $\{0,1\}$  entweder, dass sie entscheidbar ist, oder zeigen Sie mit Hilfe des Satzes von Rice, dass sie unentscheidbar ist. Geben Sie beim Beweis der Unentscheidbarkeit die Menge  $S$  der berechenbaren Funktionen an, auf die Sie den Satz von Rice anwenden. Wir bezeichnen die Länge der Eingabe  $x$  mit  $|x|$ .

- (a)  $L = \{ w \mid M_w \text{ akzeptiert die Binarkodierungen der Primzahlen (und lehnt alles andere ab)} \}$
- (b)  $L = \{ w \mid \text{es gibt eine Eingabe } x, \text{ so dass } M_w(x) \text{ das Symbol 1 enthält} \}$
- (c)  $L = \{ w \mid M_w(x) \text{ hält für jedes } x \text{ mit } |x| < 1000 \text{ nach höchstens 100 Schritten an} \}$
- (d)  $L = \{ w \mid M_w \text{ hat für jede Eingabe dieselbe Ausgabe} \}$
- (e)  $L = \{ w \mid \text{die Menge der Eingaben, die von } M_w \text{ akzeptiert werden, ist endlich} \}$

# Komplexitätstheorie

66115 / 2021 / Frühjahr / Thema 2 / Aufgabe 3

## Reduktion-Turingmaschine

Betrachten Sie das folgende Entscheidungsproblem:

Eingabe: eine geeignete codierte Turingmaschine  $M$  Ausgabe: entscheiden, ob die Turingmaschine  $M$  auf jedes Eingabewort nach höchstens 42 Schritten hält. Ist dieses Problem entscheidbar? Beweisen Sie Ihre Antwort.

Lösungsvorschlag

$M$  sei TM.  $M$  liest in jedem Schritt höchstens ein Zeichen der Eingabe.  $\Rightarrow$  Eingabe hat höchstens 42 Zeichen.  $\Rightarrow$  Menge der zu entscheidenden Wörter ist endlich.  $\Rightarrow$  Wir können alle Wörter der Sprache aufzählen und damit das Problem lösen.

Beweisen Sie mit Hilfe eines Reduktionsbeweises, dass das folgende Problem nicht entscheidbar ist:

Eingabe: zwei (geeignete codierte) Turingmaschinen  $M_1$  und  $M_2$  sowie ein Eingabewort  $\omega$  Ausgabe: entscheiden, ob  $M_1$  auf Eingabewort  $\omega$  hält und  $M_2$  auf  $\omega$  nicht hält.

Lösungsvorschlag

Das beschriebene Problem sei  $H_N$ . Die TM  $M_N$ , die zu  $H_N$  gehört, sei wie folgt definiert: • Wir wählen eine zu  $\omega$  passende TM  $M_0$  aus dem Halteproblem  $H_0$  aus, so dass  $M_0(\omega)$  hält. • Wir definieren eine TM  $M_\perp$ , die zu keiner Eingabe hält. Dann ist für  $M_N M_0(w) \# M_\perp(w)$  eine Möglichkeit für das Problem  $H_N$ . Da aber  $H_0$  nicht entscheidbar, so ist auch  $H_N$  nicht entscheidbar.

66115 / 2021 / Frühjahr / Thema 2 / Aufgabe 3

## Sat-3Sat

### Exkurs: Sat

Das **Erfüllbarkeitsproblem der Aussagenlogik** SAT und  $k$ -SAT mit  $k \geq 3, k \in \mathbb{N}$  (Satz von Cook) fragt, ob eine aussagenlogische Formel erfüllbar ist. Das Erfüllbarkeitsproblem der Aussagenlogik ist in exponentieller Zeit in Abhängigkeit der Anzahl der Variablen mit Hilfe einer Wahrheitstabelle entscheidbar. Diese *Wahrheitstabelle* kann nicht in polynomieller Zeit aufgestellt werden.

- (a) Wie zeigt man die NP-Schwere des 3SAT-Problems die NP-Schwere des SAT-Problems?

$$3\text{SAT} \preceq_p \text{SAT}$$

Jedes 3SAT-Problem ist auch ein SAT-Problem, weil  $3\text{SAT} \subset \text{SAT}$ . Damit braucht es keine Funktion (bzw. Identitäts-/Einheitsfunktion). Die Funktion ist korrekt, total und in Polynomialzeit anwendbar. Das SAT-Problem ist ebenfalls NP-schwer.

- (b) Wie zeigt man die NP-Schwere des SAT-Problems die NP-Schwere des 3SAT-Problems?

$$\text{SAT} \preceq_p 3\text{SAT}$$

Man muss eine Funktion finden, die eine allgemeine Aussagenlogik in eine Aussagenlogik mit 3 Literalen in konjunktiver Normalform umformt.

Durch die boolsche Algebra lässt sich jede logische Aussagenlogik in eine konjunktive Normalform bringen. Dies ist eine Konjunktion von Disjunktionstermen. Wir formen einen Disjunktionsterm mithilfe einer Funktion in ein 3SAT-Problem um. Diese Funktion kann auf jeden Disjunktionsterm angewendet werden und damit wird das gesamte SAT-Problem auf 3SAT reduzieren.

Die Funktion formt Formel aus SAT mithilfe von Hilfsvariablen  $h_1, \dots, h_{n-2}$  derart um  $(a_1 \vee \dots \vee a_n) \rightarrow (a_1 \vee a_2 \vee h_1) \wedge (\neg h_1 \vee a_3 \vee h_2) \wedge (\neg h_2 \vee a_4 \vee h_3) \wedge \dots \wedge (\neg h_{n-2} \vee a_n)$

**total** Diese Funktion ist total, denn jede in SAT enthaltene Aussagenlogik kann so umgewandelt werden.

**Korrektheit:** Die Hilfsvariablen sind wahr, solange bis ein Literal  $a_x$  selber true ist. Ab diesem Zeitpunkt sind die Hilfsvariablen dann falsch.

**JA-Instanzen:** Der erste und alle mittleren Disjunktionstermen sind wahr, weil aufgrund der Nicht-Negierung und Negierung immer ein wahres Literal in den Disjunktionstermen. Somit ist dann auch der Disjunktionsterm wahr. Da es eine JA-Instanz ist, existiert ein  $a_x$  welches wahr ist. Somit sind ab diesem Zeitpunkt die Hilfsvariablen falsch. Der letzte Disjunktionsterm wird dadurch sicher wahr, weil  $\neg h_{n-2}$  somit wahr ist.

**NEIN-Instanz:** Alle  $a_x$  sind falsch. Auch hier sind wieder der erste und alle mittleren Disjunktionsterme wahr (gleiche Begründung wie oben). Der letzte Disjunktionsterm ist allerdings falsch, weil die Hilfsvariablen durchgehend wahr bleiben und alle  $a_x$  falsch sind. Durch die Konjunktion der Disjunktionsterme ist dann auch die Gesamtaussage falsch.

**Polynomialzeit:** Der Algorithmus, der Formeln aus SAT nach 3SAT umformt liegt in  $\mathcal{O}(n)$  und somit in Polynomialzeit.

## 46115 / 2016 / Frühjahr / Thema 1 / Aufgabe 5

## Aufgabe 5

Beschreiben Sie, was es heißt, dass ein Problem (Sprache) NP-vollständig ist. Geben Sie ein NP-vollständiges Problem Ihrer Wahl an und erläutern Sie, dass (bzw.) warum es in NP liegt.

Lösungsvorschlag

NP-vollständig: NP-schwer und in NP

[Beliebiges Problem] liegt in NP, da der entsprechende Algorithmus dieses Problem nicht-deterministisch in Polynomialzeit löst → Algorithmus rät nichtdeterministisch Lösung, prüft sie in Polynomialzeit

## 46115 / 2016 / Frühjahr / Thema 2 / Aufgabe 2

## Aufgabe 2

Ein Raumausstattungsunternehmen steht immer wieder vor dem Problem, feststellen zu müssen, ob ein gegebener rechteckiger Fußboden mit rechteckigen Teppichresten ohne Verschnitt ausgelegt werden kann. Alle Längen sind hier ganzzahlige Meterbeiträge. Haben sie beispielsweise zwei Reste der Größen  $3 \times 5$  und einen Rest der Größe  $2 \times 5$ , so kann ein Fußboden der Größe  $8 \times 5$  ausgelegt werden.

Das Unternehmen beauftragt eine Softwarefirma mit der Entwicklung eines Programms, welches diese Frage für beliebige Größen von Fußboden und Teppichresten entscheiden soll. Bei der Abnahme weist die Softwarefirma darauf hin, dass das Programm im wesentlichen alle Möglichkeiten durchprobiert und daher für große Eingaben schnell ineffizient wird. Auf die Frage, ob man das nicht besser machen könne, lautet die Antwort, dass das vorgelegte Problem NP-vollständig sei und daher nach derzeitigem Kenntnisstand der theoretischen Informatik nicht mehr zu erwarten sei.

**Exkurs: SUBSET SUM**

Das **Teilsommenproblem** (SUBSET SUM oder SSP) ist ein spezielles Rucksackproblem. Gegeben sei eine Menge von ganzen Zahlen  $I = \{w_1, w_2, \dots, w_n\}$ . Gesucht ist eine Untermenge, deren Elementsumme maximal, aber nicht größer als eine gegebene obere Schranke  $c$  ist.

- (a) Fixieren Sie ein geeignetes Format für Instanzen des Problems und geben Sie konkret an, wie die obige Beispielinstantz in diesem Format aussieht.

Lösungsvorschlag

Problem  $L$

**1. Alternative**  $I = \{x_1, y_1, \dots, x_n, y_n, c_x, c_y\}$

**2. Alternative**  $I = \{w_1, \dots, w_n, c\}$

- (b) Begründen Sie, dass das Problem in NP liegt.

Es existiert ein nichtdeterministischer Algorithmus der das Problem in Polynomialzeit entscheidet:

- nichtdeterministisch Untermenge raten ( $\mathcal{O}(n)$ )
- Prüfe: ( $\mathcal{O}(n)$ )

**1. Alternative** Elementsumme der Produkte  $(x_i, y_i)$  aus Untermenge  $= c$

**2. Alternative** Elementsumme der Untermenge  $= c$

- (c) Begründen Sie, dass das Problem NP-schwer ist durch Reduktion vom NP-vollständigen Problem SUBSET-SUM.

SUBSET SUM  $\preceq_p$  L

**1. Alternative** Die Funktion  $f$  ersetzt jedes  $w_i$  durch  $w_i, 1$  und  $c$  durch  $c, 1$  und startet TM für  $L$

**Berechenbarkeit:** Hinzufügen von 1 für jedes Element, offensichtlich in Polynomialzeit

**Korrektheit:**  $w \in \text{SUBSET SUM} \Leftrightarrow f(w) \in L$ , offensichtlich, selbes Problem mit lediglich anders notierter Eingabe

**2. Alternative** Die Funktion  $f$  startet TM für  $L$

**Berechenbarkeit:** Identität, offensichtlich in Polynomialzeit

**Korrektheit:**  $w \in \text{SUBSET SUM} \Leftrightarrow f(w) \in L$  offensichtlich, selbes Problem

## 46115 / 2016 / Herbst / Thema 1 / Aufgabe 4

Betrachten Sie die beiden folgenden Probleme:

### VERTEX COVER

**Gegeben:** Ein ungerichteter Graph  $G = (V, E)$  und eine Zahl  $k \in \{1, 2, 3, \dots\}$

**Frage:** Gibt es eine Menge  $C \subseteq V$  mit  $|C| \leq k$ , so dass für jede Kante  $(u, v)$  aus  $E$  mindestens einer der Knoten  $u$  und  $v$  in  $C$  ist?

### VERTEX COVER 3

**Gegeben:** Ein ungerichteter Graph  $G = (V, E)$  und eine Zahl  $k \in \{3, 4, 5, \dots\}$ .

**Frage:** Gibt es eine Menge  $C \subseteq V$  mit  $|C| \leq k$ , so dass für jede Kante  $(u, v)$  aus  $E$  mindestens einer der Knoten  $u$  und  $v$  in  $C$  ist?

Geben Sie eine polynomielle Reduktion von VERTEX COVER auf VERTEX COVER 3 an und begründe anschließend, dass die Reduktion korrekt ist.

**Exkurs: VERTEX COVER**

Das **Knotenüberdeckungsproblem** (VERTEX COVER) fragt, ob zu einem gegebenen einfachen Graphen und einer natürlichen Zahl  $k$  eine Knotenüberdeckung der Größe von höchstens  $k$  existiert.

Das heißt, ob es eine aus maximal  $k$  Knoten bestehende Teilmenge  $U$  der Knotenmenge gibt, so dass jede Kante des Graphen mit mindestens einem Knoten aus  $U$  verbunden ist.

Lösungsvorschlag

$$\text{VERTEX COVER} \preceq_p \text{VERTEX COVER 3}$$

$f$  fügt vier neue Knoten hinzu, von denen jeweils ein Paar verbunden ist. Außerdem erhöht  $f$   $k$  um 2.

**Total:** Jeder Graph kann durch  $f$  so verändert werden.

**Korrektheit:** Wenn VERTEX COVER für  $k$  in  $G$  existiert, dann existiert auch VERTEX COVER mit  $k + 2$  Knoten in  $G'$ , da für den eingefügten Teilgraphen ein VERTEX COVER mit  $k = 2$  existiert.

**In Polynomialzeit berechenbar:** Für die Adjazenzmatrix des Graphen müssen lediglich vier neue Spalten und Zeilen eingefügt werden und  $k + 2$  berechnet werden.

## 66115 / 2014 / Frühjahr / Thema 1 / Aufgabe 7

- (a) Sei SAT das Erfüllbarkeitsproblem, und sei  $H_m$ , das Halteproblem bei fester Eingabe  $m$  (siehe Aufgabe 5).

Zeigen Sie: SAT kann in polynomieller Zeit auf  $H_m$  reduziert werden. Als Relation:

$$\text{SAT} \preceq_p H_m$$

- (b) Angenommen, es wurde gezeigt, dass  $P = NP$  ist. Zeigen Sie, dass dann jede Sprache  $L \in P$  über dem Alphabet  $\Sigma$  mit  $L \neq \emptyset$  und  $L \neq \Sigma^*$  sogar NP-vollständig ist.

## 66115 / 2016 / Frühjahr / Thema 1 / Aufgabe 5

Das Problem k-COL ist wie folgt definiert:

**K-COL****Gegeben:** Ein ungerichteter Graph  $G = (V, E)$ .**Frage:** Kann man jedem Knoten  $v$  in  $V$  eine Zahl  $z(v) \in \{1, \dots, k\}$  zuordnen, so dass für alle Kanten  $(u_1, u_2) \in E$  gilt:  $z(u_1) \neq z(u_2)$ ?

Zeigen Sie, dass man 3-COL in polynomieller Zeit auf 4-COL reduzieren kann. Beschreiben Sie dazu die Reduktion und zeigen Sie anschließend ihre Korrektheit.

Lösungsvorschlag

Zu Zeigen:

$$3\text{-COL} \preceq_p 4\text{-COL}$$

also 4-COL ist mindestens so schwer wie 3-COL Eingabeinstanz von 3-COL durch eine Funktion in eine Eingabeinstanz von 4-COL umbauen so, dass jede JA- bzw. NEIN-Instanz von 3-COL eine JA- bzw. NEIN-Instanz von 4-COL ist.

Funktion ergänzt einen beliebigen gegebenen Graphen um einen weiteren Knoten, der mit allen Knoten des ursprünglichen Graphen durch eine Kante verbunden ist.

**total** ja

**in Polynomialzeit berechenbar** ja (Begründung: z. B. Adjazenzmatrix  $\rightarrow$  neue Spalte)

**Korrektheit:** ja

Färbe den „neuen“ Knoten mit einer Farbe. Da er mit allen anderen Knoten verbunden ist, bleiben für die übrigen Knoten nur drei Farben.

**66115 / 2016 / Frühjahr / Thema 2 / Aufgabe 3**

Beantworten Sie kurz, präzise und mit Begründung folgende Fragen: (Die Begründungen müssen keine formellen mathematischen Beweise sein)

- (a) In der O-Notation insbesondere für die Zeitkomplexität von Algorithmen lässt man i. A. konstante Faktoren oder kleinere Terme weg. Z. B. schreibt man anstelle  $\mathcal{O}(3n^2 + 5)$  einfach nur  $\mathcal{O}(n^2)$ . Warum macht man das so?

Lösungsvorschlag

Das Wachstum im Unendlich ist bestimmt durch den größten Exponenten. Konstante fälle bei einer asymptotischen. Analyse weg. nicht wesentlich schneller

- (b) Was ist die typische Vorgehensweise, wenn man für ein neues Problem die NP-Vollständigkeit untersuchen will?



Die alten Probleme werden reduziert. Das neue Problem ist größer als die alten Probleme. Das Problem muss in NP liegen.

- (i) Problem *in* NP durch Angabe eines nichtdeterministischen Algorithmus in Polynomialzeit
- (ii) Problem NP-schwer via Reduktion:  $L_{\text{NP-vollständig}} \leq L_{\text{neues Problem}}$

(c) Was könnte man tun, um  $P = NP$  zu beweisen?

Lösungsvorschlag

Es würde genügen, zu einem einzigen NP-Problem beweisen, dass es in P liegt.

Zu einem Problem einen deterministischen Turingmaschin finden, die es in polynomineller Zeit löst.

(d) Sind NP-vollständige Problem mit Loop-Programmen lösbar? (Antwort mit Begründung!)

Lösungsvorschlag

nicht lösbar mit Loop-Programmen. Begründung ähnlich wie bei der Ackermann-Funktion. z. B. Passwort. Zu Passwort beliebig bräuchte man beliebige for schleifen, was dem endlichen Anzahl an Loop-Schleifen widerspricht.

(e) Wie zeigt man aus der NP-Härte des SAT-Problems die NP-Härte des 3SAT-Problems? (3SAT ist ein SAT-Problem wobei alle Klauseln maximal 3 Literale haben.)

Lösungsvorschlag

in den Lösungen enthalten

## 66115 / 2017 / Herbst / Thema 1 / Aufgabe 3

Betrachten Sie die folgenden Probleme:

### Exkurs: SAT

Das **Erfüllbarkeitsproblem der Aussagenlogik** SAT und  $k$ -SAT mit  $k \geq 3, k \in \mathbb{N}$  (Satz von Cook) fragt, ob eine aussagenlogische Formel erfüllbar ist. Das Erfüllbarkeitsproblem der *Aussagenlogik* ist in exponentieller Zeit in Abhängigkeit der Anzahl der Variablen mit Hilfe einer Wahrheitstabelle entscheidbar. Diese *Wahrheitstabelle* kann nicht in polynomieller Zeit aufgestellt werden.

### Exkurs: NAE3SAT

Like 3-satisfiability, an instance of the problem consists of a collection of Boolean variables and a collection of clauses, each of which combines three variables or negations of variables. However, unlike 3-satisfiability, which requires each clause to have at least one true Boolean value, NAE3SAT requires that the three values in each clause are not all equal to each other (in other words, at least

one is true, and at least one is false)<sup>a</sup>

<sup>a</sup>[https://en.wikipedia.org/wiki/Not-all-equal\\_3-satisfiability](https://en.wikipedia.org/wiki/Not-all-equal_3-satisfiability)

### 3SAT

**Gegeben:** Eine aussagenlogische Formel  $\varphi$  in konjunktiver Normalform (drei Literale pro Klausel).

**Frage:** Ist  $\varphi$  erfüllbar?

### NAE-3SAT

**Gegeben:** Eine aussagenlogische Formel  $\varphi$  in konjunktiver Normalform (drei Literale pro Klausel).

**Frage:** Gibt es eine Belegung, die in jeder Klausel mindestens ein Literal *wahr* und mindestens ein Literal *falsch* macht?

Wir erlauben, dass NAE-3SAT-Formeln Literale der Form *false* haben, die immer *falsch* sind. So ist

$$(x_1 \vee \text{false} \vee \text{false}) \wedge (\neg x_1 \vee x_1 \vee x_1)$$

in NAE-3SAT (setze  $x_1$  wahr).

- (a) Zeigen Sie, dass sich 3SAT in polynomieller Zeit auf NAE-3SAT reduzieren lässt.

Lösungsvorschlag

- (b) Was können Sie aus a) folgern, wenn Sie wissen, dass 3SAT NP-vollständig ist?

Lösungsvorschlag

- (c) Was können Sie aus a) folgern, wenn Sie wissen, dass NAE-3SAT NF-vollständig ist?

Lösungsvorschlag

## 66115 / 2018 / Herbst / Thema 2 / Aufgabe 4

Wir betrachten ungerichtete Graphen  $G = (V, E)$ , wo  $E$  eine Teilmenge  $E$  hat, die wir exklusive Kanten nennen. Eine beschränkte Überdeckung von  $G$  ist eine Teilmenge  $U$  von  $V$ , so dass

- (a) jeder Knoten einen Nachbarknoten in  $U$  hat oder selbst in  $U$  liegt (für jeden Knoten  $u \in V \setminus U$  gibt es einen Knoten  $v \in U$  mit  $(u, v) \in E$ ) und
- (b) für jede exklusive Kante  $(u, v) \in E$  genau einer der Knoten  $u, v$  in  $U$  liegt.

Betrachten Sie nun die folgenden Entscheidungsprobleme:

**3SAT****Gegeben:** Aussagenlogische Formel  $p$  in 3KNF**Frage:** Hat  $p$  eine erfüllende Belegung?**BÜ****Gegeben:** Graph  $G = (V, E)$ , exklusive Kantenmenge  $E$ .  $C \subseteq V$  und  $k \in \mathbb{N}$ **Frage:** Hat  $G$  eine beschränkte Überdeckung  $U$  mit  $|U| \leq k$ ?

Beweisen Sie, dass BÜ NP-vollständig ist. Sie dürfen dabei annehmen, dass 3SAT NP-vollständig ist.

**66115 / 2019 / Frühjahr / Thema 1 / Aufgabe 5**

Wir betrachten das Behälterproblem BEHAELTER. Gegeben ist eine Menge von  $k \in \mathbb{N}$  Behältern, die jeweils ein Fassungsvermögen der Größe  $b \in \mathbb{N}$  haben. Gegeben sind weiterhin  $n$  Objekte mit jeweiligen Größen  $a_1, \dots, a_n$ . Gesucht ist eine Zuordnung der  $n$  Objekte auf die  $k$  Behälter, sodass keiner der Behälter überläuft.

Formal sind Instanzen des Behälterproblems BEHAELTER durch Tupel  $(k, a_1, \dots, a_n)$  gegeben, die wie folgt zu interpretieren sind:

- $k \in \mathbb{N}$  steht für eine Anzahl von Behältern.
- Jeder Behälter hat ein Fassungsvermögen von  $b \in \mathbb{N}$ .
- Die  $a_i$  stehen für die jeweiligen Größen von  $n$  Objekten.

Zuordnungen von Objekten zu Behältern geben wir durch eine Funktion  $v$  an, wobei  $v(j) = i$  wenn das  $j$ -te Objekt (mit Größe  $a_j$ ) dem  $i$ -ten Behälter zugeordnet wird.

$(k, b, a_1, \dots, a_n)$  ist eine JA-Instanz von BEHAELTER, wenn es eine Zuordnung  $v$  von Objekten auf Behälter ( $v : [1; n] \rightarrow [1; k]$ ) gibt, die sicherstellt, dass kein Behälter überläuft:

$$(k, b, a_1, \dots, a_n) \in \text{BEHAELTER} \iff (\exists v : [1; n] \rightarrow [1; k]. \forall i \in [1; k]. \sum_{j: v(j)=i} a_j \leq b)$$

Wir betrachten auch das modifizierte Problem GERADEBEHAELTER. Instanzen von GERADEBEHAELTER tragen die zusätzliche Einschränkung, dass alle  $a_i$  gerade (durch zwei teilbar) sein müssen.

- (a) Warum ist sowohl BEHAELTER  $\in$  NP als auch GERADEBEHAELTER  $\in$  NP?
- (b) Beweisen Sie, dass das Problem BEHAELTER auf das Problem GERADEBEHAELTER in polynomieller Zeit reduzierbar ist.
- (c) BEHAELTER ist NP-vollständig. Begründen Sie, was obige Reduktion für die Komplexität von GERADEBEHAELTER bedeutet. BEHAELTER ist NP-vollständig. Begründen Sie, was obige Reduktion für die Komplexität von GERADEBEHAELTER bedeutet.

## 66115 / 2019 / Herbst / Thema 1 / Aufgabe 4

Betrachten Sie die folgenden Probleme:

**SAT**

**Gegeben:** Aussagenlogische Formel  $F$  in KNF

**Frage:** Gibt es mindestens eine erfüllende Belegung für  $F$ ?

**DOPPELSAT**

**Gegeben:** Aussagenlogische Formel  $F'$  in KNF

**Frage:** Gibt es mindestens eine erfüllende Belegung für  $F$ , in der mindestens zwei Literale pro Klausel wahr sind?

- (a) Führen Sie eine polynomielle Reduktion von SAT auf DOPPELSAT durch.

Lösungsvorschlag

<https://courses.cs.washington.edu/courses/csep531/09wi/handouts/sol4.pdf>

DOUBLE-SAT is in NP. The polynomial size certificate consists of two assignments  $f_1$  and  $f_2$ . First, the verifier verifies if  $f_1 \neq f_2$ . Then, it verifies if both assignments satisfy  $\phi$  by substituting the values for the variables and evaluate the clauses of  $\phi$ . Both checks can be done in linear time. DOUBLE-SAT is NP-hard. We give a reduction from SAT. Given an instance  $\phi$  of SAT which is a CNF formula of  $n$  variables  $x_1, x_2, \dots, x_n$ , we construct a new variable  $x_{n+1}$  and let  $\psi = \phi \wedge (x_{n+1} \vee \neg x_{n+1})$  be the corresponding instance of DOUBLE-SAT. We claim that  $\phi$  has a satisfying assignment iff  $\psi$  has at least two satisfying assignments. On one hand, if  $\phi$  has a satisfying assignment  $f$ , we can obtain two distinct satisfying assignments of  $\psi$  by extending  $f$  with  $x_{n+1} = T$  and  $x_{n+1} = F$  respectively. On the other hand, if  $\psi$  has at least two satisfying assignments then the restriction of any of them to the set  $x_1, x_2, \dots, x_n$  is a satisfying assignment for  $\phi$ . Thus, DOUBLE-SAT is NP-complete.

<https://cs.stackexchange.com/questions/6371/proving-double-sat-is-np-complete>

Here is one solution:

Clearly Double-SAT belongs to NP, since a NTM can decide Double-SAT as follows: On a Boolean input formula  $\phi(x_1, \dots, x_n)$ , nondeterministically guess 2 assignments and verify whether both satisfy  $\phi$ .

To show that Double-SAT is NP-Complete, we give a reduction from SAT to Double-SAT, as follows:

On input  $\phi(x_1, \dots, x_n)$ :

1. Introduce a new variable  $y$ .
2. Output formula  $\phi'(x_1, \dots, x_n, y) = \phi(x_1, \dots, x_n) \wedge (y \vee \bar{y})$ .

If  $\phi(x_1, \dots, x_n)$  belongs to SAT, then  $\phi$  has at least 1 satisfying assignment, and therefore  $\phi'(x_1, \dots, x_n, y)$  has at least 2 satisfying assignments as we can satisfy the new clause  $(y \vee \bar{y})$  by assigning either  $y = 1$  or  $y = 0$  to the new variable  $y$ , so  $\phi'(x_1, \dots, x_n, y) \in \text{Double-SAT}$ .

On the other hand, if  $\phi(x_1, \dots, x_n) \notin \text{SAT}$ , then clearly  $\phi'(x_1, \dots, x_n, y) = \phi(x_1, \dots, x_n) \wedge (y \vee \bar{y})$  has no satisfying assignment either, so  $\phi'(x_1, \dots, x_n, y) \notin \text{Double-SAT}$ .

Therefore,  $\text{SAT} \leq_p \text{Double-SAT}$ , and hence Double-SAT is NP-Complete.

(b) Zeigen Sie, dass DOPPELSAT NP-vollständig ist.

### 66115 / 2020 / Herbst / Thema 1 / Teilaufgabe 1 / Aufgabe 5

Sei  $\mathcal{IF}$  die Menge aller aussagenlogischen Formeln, die ausschließlich mit den Konstanten 0 und 1, logischen Variablen  $x_i$  mit  $i \in \mathbb{N}$  und der Implikation  $\Rightarrow$  als Operationszeichen aufgebaut sind, wobei natürlich Klammern zugelassen sind. Beachten Sie, dass  $x_i \Rightarrow x_j$  die gleiche Wahrheitstabelle wie  $\neg x_i \vee x_j$  hat.

Wir betrachten das Problem  $\text{ISAT}$ . Eine Formel  $F \in \mathcal{IF}$  ist genau dann in  $\text{ISAT}$  enthalten, wenn sie erfüllbar ist, das heißt, falls es eine Belegung der Variablen mit Konstanten 0 oder 1 gibt, sodass  $F$  den Wert 1 annimmt.

Zeigen Sie:  $\text{ISAT}$  ist NP-vollständig. Sie dürfen benutzen, dass das SAT-Problem NP-vollständig ist.

Lösungsvorschlag

### 66115 / 2020 / Herbst / Thema 2 / Teilaufgabe 2 / Aufgabe 2

Beweisen Sie die folgenden Aussagen:

(a) Sei  $f(n) = 2n^2 + 3n + 4 \log(n) + 7$ . Dann gilt  $f \in O(n^2)$ .

(b) Sei  $f(n) = 4^n$ . Dann gilt nicht  $f \in O(2^n)$ .

Lösungsvorschlag

(c) Sei  $f(n) = (n+1)!$  (d. h. die Fakultät von  $n+1$ ). Dann gilt  $f \in O(n^n)$ .

Lösungsvorschlag

z. B.  $5 \cdot 4 \cdot 3 \cdot 2 \cdot 1 \cdot 5^5$

(d) Sei  $f: \mathbb{N} \leftarrow \mathbb{N}$  definiert durch die folgende Rekursionsgleichung:

$$f(n) = \begin{cases} 3, & \text{für } n = 1 \\ (n-1)^2 + f(n-1), & \text{für } n > 1 \end{cases}$$

Dann gilt  $f \in O(n^3)$

$$\begin{aligned}
 f(n) &= (n-1)^2 + f(n-1) + \dots + f(1) \\
 &= (n-1)^2 + (n-1)^2 + f(n-2) + \dots + f(1) \\
 &= \underbrace{(n-1)^2 + \dots + (n-1)^2}_n + 3 \\
 &= \underbrace{(n-1)^2 + \dots + (n-1)^2}_{n-1} + 3 \\
 &= (n-1)^2 \cdot (n-1) + 3 \\
 &= (n-1)^3 + 3
 \end{aligned}$$

## 66115 / 2021 / Frühjahr / Thema 2 / Teilaufgabe 1 / Aufgabe 4

Betrachten Sie die folgenden Probleme:

### CLIQUE

**Gegeben:** Ein ungerichteter Graph  $G = (V, E)$ ,  
eine Zahl  $k \in \mathcal{N}$

**Frage:** Gibt es eine Menge  $S \subseteq V$  mit  $|S| = k$ ,  
sodass für alle Knoten  $u \neq v \in V$  gilt,  
dass  $\{u, v\}$  eine Kante in  $E$  ist?

### ALMOST CLIQUE

**Gegeben:** Ein ungerichteter Graph  $G = (V, E)$ ,  
eine Zahl  $k \in \mathcal{N}$

**Frage:** Gibt es eine Menge  $S \subseteq V$  mit  $|S| = k$ ,  
sodass die Anzahl der Kanten zwischen Knoten in  $S$  genau  $\frac{k(k-1)}{2} - 1$  ist?

Zeigen Sie, dass das Problem ALMOST CLIQUE NP-vollständig ist. Nutzen Sie dafür die NP-Vollständigkeit von CLIQUE.

Hinweis: Die Anzahl der Kanten einer  $k$ -Clique sind  $\frac{k(k-1)}{2}$ .

### Exkurs: Cliquesproblem

Das **Cliquesproblem** fragt nach der Existenz einer Clique der Mindestgröße  $n$  in einem gegebenen Graphen. Eine Clique ist eine Teilmenge von Knoten in einem ungerichteten Graphen, bei der *jedes Knotenpaar durch eine Kante verbunden ist*.

**Exkurs: Almost Clique**

Eine Gruppe von Knoten wird `ALMOST CLIQUE` genannt, wenn nur eine Kante ergänzt werden muss, damit sie zu einer Clique wird.

Lösungsvorschlag

You can reduce to this from *CLIQUE*.

Given a graph  $G = (V, E)$  and  $t$ , construct a new graph  $G^*$  by adding two new vertices  $\{v_{n+1}, v_{n+2}\}$  and connecting them with all of  $G$ 's vertices but removing the edge  $\{v_{n+1}, v_{n+2}\}$ , i.e. they are not neighbors in  $G^*$ . return  $G^*$  and  $t + 2$ .

If  $G$  has a  $t$  sized clique by adding it to the two vertices we get an  $t + 2$  almost clique in  $G^*$  (by adding  $\{v_{n+1}, v_{n+2}\}$ ).

If  $G^*$  has a  $t + 2$  almost clique we can look at three cases:

1) It contains the two vertices  $\{v_{n+1}, v_{n+2}\}$ , then the missing edge must be  $\{v_{n+1}, v_{n+2}\}$  and this implies that the other  $t$  vertices form a  $t$  clique in  $G$ .

2) It contains one of the vertices  $\{v_{n+1}, v_{n+2}\}$ , say w.l.o.g.  $v_{n+1}$ , then the missing edge must be inside  $G$ , say  $e = \{u, v\} \in G$ . If we remove  $u$  and  $v_{n+1}$  then the other  $t$  vertices, which are in  $G$  must form a clique of size  $t$ .

3) It does not contain any of the vertices  $\{v_{n+1}, v_{n+2}\}$ , then it is clear that this group is in  $G$  and must contain a clique of size  $t$ .

It is also clear that the reduction is in polynomial time, actually in linear time, log-space. <sup>a</sup>

---

<sup>a</sup><https://cs.stackexchange.com/a/76627>

## Sonstige



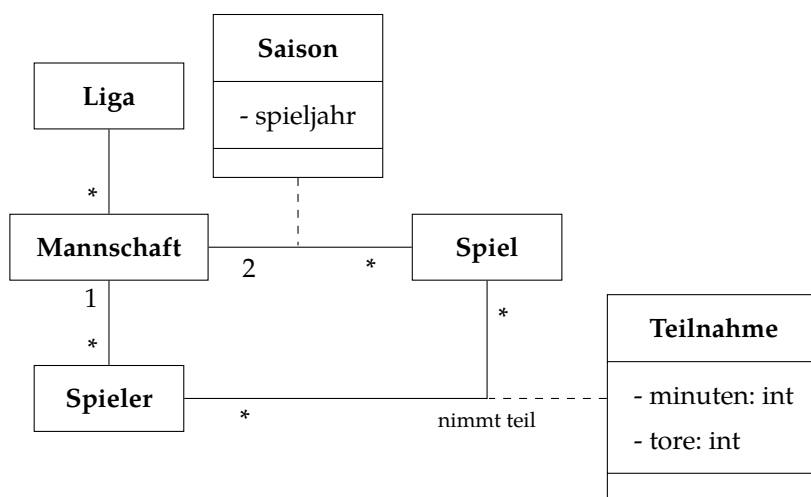
# SOSY

## 66115 / 2021 / Frühjahr / Thema 2 / Aufgabe 4

Modellieren Sie die folgende Situation als Klassendiagramm:

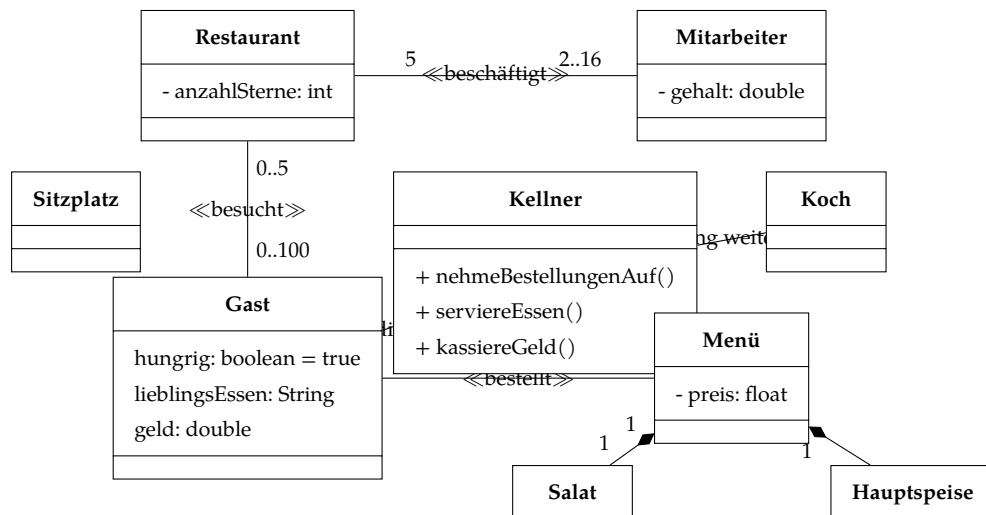
Fußballmannschaften einer Liga bestreiten während einer Meisterschaftsrunde Spiele gegen andere Mannschaften. Dabei werden in jeder Mannschaft Spieler für einen bestimmten Zeitraum (in Minuten) eingesetzt, die dabei eventuell Tore schießen.

Die Modellierung soll es ermöglichen, festzustellen, welcher Spieler in welchem Spiel wie lange auf dem Feld war und wie viele Tore geschossen hat. Ebenso soll es möglich sein, für jede Mannschaft festzustellen, gegen welche Mannschaft welche Ergebnisse erzielt wurden.



## 66115 / 2021 / Frühjahr / Thema 2 / Aufgabe 4

In diesem kleinen Städtchen gibt es fünf Restaurants, aber egal, ob es sich um ein 2\*-Restaurant, oder um ein 5\*-Restaurant handelt, alle haben Gemeinsamkeiten. In jedem Restaurant sind mehrere **Mitarbeiter** angestellt. Im kleinsten Restaurant nur zwei, aber im größten verdienen 16 Mitarbeiter ihr Geld. Die Mitarbeiter bekommen in jedem Restaurant ein anderes *Gehalt*, aber alle machen ihre Arbeit. Die **Kellner** nehmen *bestellungen entgegen*, *servieren das Essen* und *kassieren das Geld* von den Gästen. Jeder Kellner gibt die Bestellungen der Gäste an die Köche weiter. Die **Köche** kennen verschiedene **Rezepte**, nach denen sie die **Menüs** dann zubereiten. Jedes Menü hat einen anderen *Preis*, aber es besteht nach Tradition immer aus einem **Salat** und einer **Hauptspeise**. An manchen Tagen ist kein **Gast** da. Dann werden Sie in ganz familiärem Rahmen bewirtet. Aber auch an guten Tagen gibt es, bei 100 Plätzen im größten Restaurant, sicher für jeden hungrigen Gast einen **Platz**. Wer in Gasthausen Essen geht sollte hungrig sein! Natürlich hat jeder Gast unterschiedlich viel Geld zur Verfügung, aber bei entsprechender Wahl des Restaurants reicht es sicher für sein Lieblingsessen.



## 66115 / 2021 / Frühjahr / Thema 2 / Aufgabe 4

(a) Erstelle Sie ein Klassendiagramm zu folgender Aufgabenstellung:

Es gibt eine Klasse **Hund**, von der keine Objekte erzeugt werden sollen. Jeder Hund hat einen Namen (**String**), ein Alter in Jahren (**int**) und ein Gewicht in kg (**double**). Jeder Hund kann bellen, älter werden (um jeweils 1 Jahr), eine gewisse Menge an Hundefutter fressen und dabei schwerer werden, und Gassi gehen, wobei er wieder an Gewicht verliert. Hunde sind entweder Chihuahuas, Bernhardiner oder Schäferhunde. Chihuahuas bellen mit einem wuffwuff, wiegen durchschnittlich 2 kg, fressen maximal 0,09 kg pro Tag und verlieren ca. 0,04 kg pro Gassi-Gang. Ein Bernhardiner bellt mit einem lauten WAUWAU, wiegt durchschnittlich 95 kg, frisst maximal 0,5 kg pro Tag und verliert pro Gassi-Gang ca. 0,2 kg. Bernhardiner tragen eine Glocke um den Hals oder nicht. Ein Schäferhund bellt mit einem wauwau, wiegt durchschnittlich 30 kg, frisst maximal 0,3 kg pro Tag und verliert ca. 0,15 kg pro Gassi-Gang. Ein Schäferhund hat die Ausbildung zum Blindenhund absolviert oder nicht.

(b) Erstellen Sie nun zunächst die Klasse **Hund** mit den oben angegebenen Attributen und folgenden Methoden:

- Methode `bellen()`;
- Methode `altern()`;
- Methode `fressen(double futter)`;
- Methode `fressen()`: Verwendet als Futtermenge die maximale Menge pro Tag
- Methode `gassiGehen()`;

(c) Erstellen Sie nun die Klassen **Chihuahua**, **Bernhardiner** und **Schaeferhund** mit ihren spezifischen Attributen und überschreiben Sie die in Teilaufgabe b) genannten Methoden.

- (d) Erstellen Sie alle benötigten Getter- und Setter-Methoden.
- (e) Erstellen Sie in einer Klasse `Zwinger` ein Hunde-Array `zwinger`, das Platz für zehn Hunde-Objekte hat, und folgende Methoden:
- Eine Methode `belegen()`, die die folgenden vier Hunde in die ersten vier Zwingerplätze setzt:
    - Chihuahua Tim, 2 Jahre alt, 1,8 kg schwer
    - Blindenhund Alex, 4 Jahre alt, 40 kg schwer
    - Bernhardinerin Eva, 5 Jahre alt, 82 kg schwer
    - Schäferhündin Lilli, 3 Jahre alt, 34 kg schwer
  - Eine Methode `fuettern()`, die alle Hunde im Zwinger mit ihrer maximalen Futtermenge versorgt.
  - Eine Methode `fuetterzeit()`, die alle Hunde im Zwinger bellen lässt.
  - Eine Methode `gassiGehen()`, die alle Hunde im Zwinger Gassi gehen lässt.

```
public abstract class Hund {
    protected String name;
    protected int alter;
    protected double gewicht;

    public Hund(String n, int a, double g) {
        name = n;
        alter = a;
        gewicht = g;
    }

    public abstract void bellen();

    public void altern() {
        alter = alter + 1;
    }

    public void fressen(double futter) {
        gewicht = gewicht + futter;
    }

    public abstract void fressen();

    public abstract void gassiGehen();

    public String getName() {
        return name;
    }

    public int getAlter() {
        return alter;
    }

    public double getGewicht() {
        return gewicht;
    }
}
```

```
}

public void setName(String n) {
    name = n;
}

public void setAlter(int a) {
    if (a >= 0) {
        alter = a;
    }
}

public void setGewicht(double g) {
    if (g > 0) {
        gewicht = g;
    }
}
}
```

Code-Beispiel auf Github ansehen: <src/main/java/org/bschlangaul/aufgaben/ooup/klassendiagramm/hunde/Hund.java>

```
public class Bernhardiner extends Hund {
    private boolean glocke;

    public Bernhardiner(String n, int a, double g, boolean gl) {
        super(n, a, g);
        glocke = gl;
    }

    public void bellen() {
        System.out.println("WAUWAU");
    }

    public void fressen(double futter) {
        if (futter > 0 && futter < 0.5)
            super.fressen(futter);
    }

    public void fressen() {
        super.gewicht = super.gewicht + 0.5;
    }

    public void gassiGehen() {
        super.gewicht = super.gewicht - 0.2;
    }

    public boolean getGlocke() {
        return glocke;
    }

    public void setGlocke(boolean g) {
        glocke = g;
    }
}
```

Code-Beispiel auf Github ansehen: <src/main/java/org/bschlangaul/aufgaben/ooup/klassendiagramm/hunde/Bernhardiner.java>

```
public class Chihuahua extends Hund {
    public Chihuahua(String n, int a, double g) {
        super(n, a, g);
    }

    public void bellen() {
        System.out.println("wuffwuff");
    }

    public void fressen(double futter) {
        if (futter > 0 && futter < 0.09)
            super.fressen(futter);
    }

    public void fressen() {
        super.gewicht = super.gewicht + 0.09;
    }

    public void gassiGehen() {
        super.gewicht = super.gewicht - 0.04;
    }
}
```

Code-Beispiel auf Github ansehen: <src/main/java/org/bschlangaul/aufgaben/oomup/klassendiagramm/hunde/Chihuahua.java>

```
public class Schaeferhund extends Hund {
    private boolean blindenhund;

    public Schaeferhund(String n, int a, double g, boolean bl) {
        super(n, a, g);
        blindenhund = bl;
    }

    public void bellen() {
        System.out.println("wauwau");
    }

    public void fressen(double futter) {
        if (futter > 0 && futter < 0.3)
            super.fressen(futter);
    }

    public void fressen() {
        super.gewicht = super.gewicht + 0.3;
    }

    public void gassiGehen() {
        super.gewicht = super.gewicht - 0.15;
    }

    public boolean getBlind() {
        return blindenhund;
    }

    public void setBlind(boolean b) {
        blindenhund = b;
    }
}
```

```
}  
}
```

Code-Beispiel auf Github ansehen: [src/main/java/org/bschlangaul/aufgaben/oomup/klassendiagramm/hunde/Schaeferhund.java](https://github.com/bschlangaul/aufgaben/oomup/klassendiagramm/hunde/Schaeferhund.java)

```
public class Zwinger {  
    // Attribute  
    private Hund[] zwinger;  
  
    // Konstruktor  
    public Zwinger() {  
        zwinger = new Hund[10];  
    }  
  
    // Methoden  
    public void belegen() {  
        zwinger[0] = new Chihuahua("Tim", 2, 1.8);  
        zwinger[1] = new Schaeferhund("Alex", 4, 40.0, true);  
        zwinger[2] = new Bernhardiner("Eva", 5, 82.0, false);  
        zwinger[3] = new Schaeferhund("Lilli", 3, 34.0, false);  
    }  
  
    public void fuettern() {  
        for (int i = 0; i < zwinger.length; i++) {  
            if (zwinger[i] != null) {  
                zwinger[i].fressen();  
            }  
        }  
    }  
  
    public void fuetterzeit() {  
        for (int i = 0; i < zwinger.length; i++) {  
            if (zwinger[i] != null) {  
                zwinger[i].bellen();  
            }  
        }  
    }  
  
    public void gassiGehen() {  
        for (int i = 0; i < zwinger.length; i++) {  
            if (zwinger[i] != null) {  
                zwinger[i].gassiGehen();  
            }  
        }  
    }  
}
```

Code-Beispiel auf Github ansehen: [src/main/java/org/bschlangaul/aufgaben/oomup/klassendiagramm/hunde/Zwinger.java](https://github.com/bschlangaul/aufgaben/oomup/klassendiagramm/hunde/Zwinger.java)

## 66115 / 2021 / Frühjahr / Thema 2 / Aufgabe 4

In der Kleintierpraxis Dr. Huf werden *Hunde*, *Katzen* und *Heimtiere*<sup>1</sup> behandelt. In der Praxissoftware werden von jedem Tier *Name*, *Alter* (in Jahren) und *Gewicht* (in kg mit

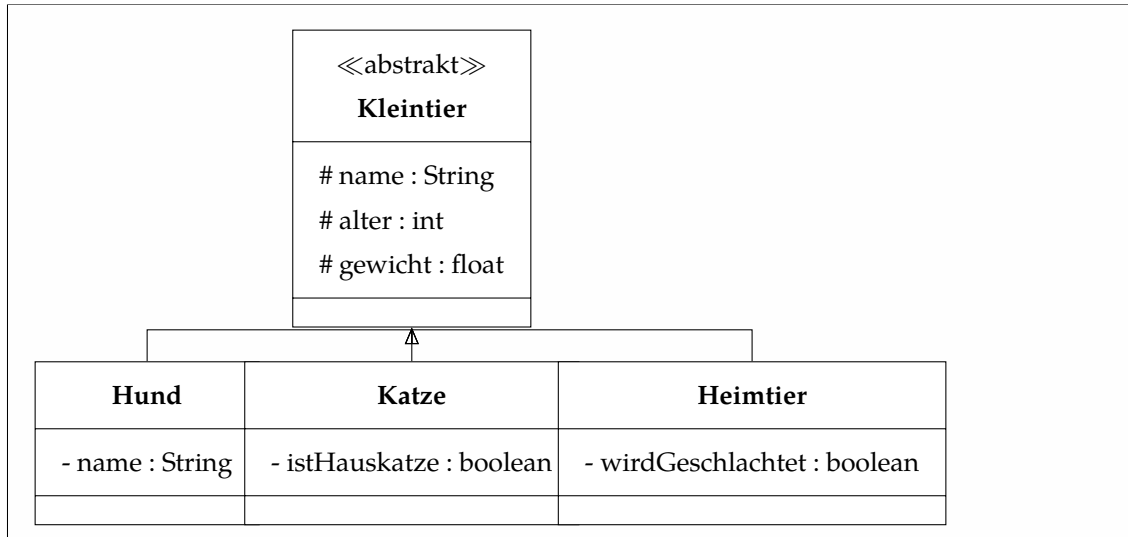
---

<sup>1</sup>Unter Heimtiere versteht man hier Kleintiere (d. h. keine Rinder, Pferde etc.), die zu Hause gehalten werden und keine Hunde oder Katzen sind (beispielsweise Meerschweinchen oder Kaninchen). Heim-

mindestens 2 Dezimalen) erfasst. Bei einem *Hund* wird zusätzlich aufgenommen, ob er *reinrassig* ist, bei einer *Katze*, ob sie ausschließlich *in der Wohnung gehalten* wird, und beim *Heimtier*, ob es zur *Lebensmittellieferung* dient.

- (a) Erstellen Sie ein entsprechendes Klassendiagramm (zunächst ohne Methoden), das den oben beschriebenen Sachverhalt geeignet erfasst.

Lösungsvorschlag



- (b) Implementieren Sie die Klassen gemäß des Modells aus Teilaufgabe a in der Programmierumgebung BlueJ. Beachten Sie dabei die abstrakte Oberklasse!

Lösungsvorschlag

### Klasse *Kleintier*

```
public abstract class Kleintier {
    protected String name;
    protected int alter;
    protected float gewicht;
}
```

Code-Beispiel auf Github ansehen: [src/main/java/org/bschlangaul/aufgaben/oomup/klassendiagramm/kleintierpraxis/Kleintier.java](https://github.com/bschlangaul/aufgaben/oomup/klassendiagramm/kleintierpraxis/Kleintier.java)

```
public Kleintier(String name, int alter, float gewicht) {
    this.name = name;
    this.alter = alter;
    this.gewicht = gewicht;
}
```

Code-Beispiel auf Github ansehen: [src/main/java/org/bschlangaul/aufgaben/oomup/klassendiagramm/kleintierpraxis/Kleintier.java](https://github.com/bschlangaul/aufgaben/oomup/klassendiagramm/kleintierpraxis/Kleintier.java)

### Klasse *Hund*

tiere können auch Nutztiere sein, die zur Schlachtung gehalten werden (z. B. Hühner). In diesem Fall muss ein Tierarzt darauf achten, nur Medikamente anzuwenden, die für Lebensmittel liefernde Tiere zugelassen sind.

```
public class Hund extends Kleintier {
    private boolean reinrassig;

    public Hund(String name, int alter, float gewicht, boolean reinrassig) {
        super(name, alter, gewicht);
        this.reinrassig = reinrassig;
    }
}
```

Code-Beispiel auf Github ansehen: [src/main/java/org/bschlangaul/aufgaben/oomup/klassendiagramm/kleintierpraxis/Hund.java](https://github.com/bschlangaul/aufgaben/oomup/klassendiagramm/kleintierpraxis/Hund.java)

### Klasse *Katze*

```
public class Katze extends Kleintier {
    @SuppressWarnings("unused")
    private boolean istHauskatze;

    public Katze(String name, int alter, float gewicht, boolean istHauskatze) {
        super(name, alter, gewicht);
        this.istHauskatze = istHauskatze;
    }
}
```

Code-Beispiel auf Github ansehen: [src/main/java/org/bschlangaul/aufgaben/oomup/klassendiagramm/kleintierpraxis/Katze.java](https://github.com/bschlangaul/aufgaben/oomup/klassendiagramm/kleintierpraxis/Katze.java)

### Klasse *Heimtier*

```
public class Heimtier extends Kleintier {
    @SuppressWarnings("unused")
    private boolean wirdGeschlachtet;

    public Heimtier(String name, int alter, float gewicht, boolean
    ↪ wirdGeschlachtet) {
        super(name, alter, gewicht);
        this.wirdGeschlachtet = wirdGeschlachtet;
    }
}
```

Code-Beispiel auf Github ansehen: [src/main/java/org/bschlangaul/aufgaben/oomup/klassendiagramm/kleintierpraxis/Heimtier.java](https://github.com/bschlangaul/aufgaben/oomup/klassendiagramm/kleintierpraxis/Heimtier.java)

- (c) Die Praxissoftware verfügt über eine Methode `datenAusgeben()`, die den Namen und das Alter eines Tieres auf dem Bildschirm ausgibt. Fügen Sie im Klassendiagramm die Methode an geeigneter Stelle ein und implementieren Sie sie.

Lösungsvorschlag

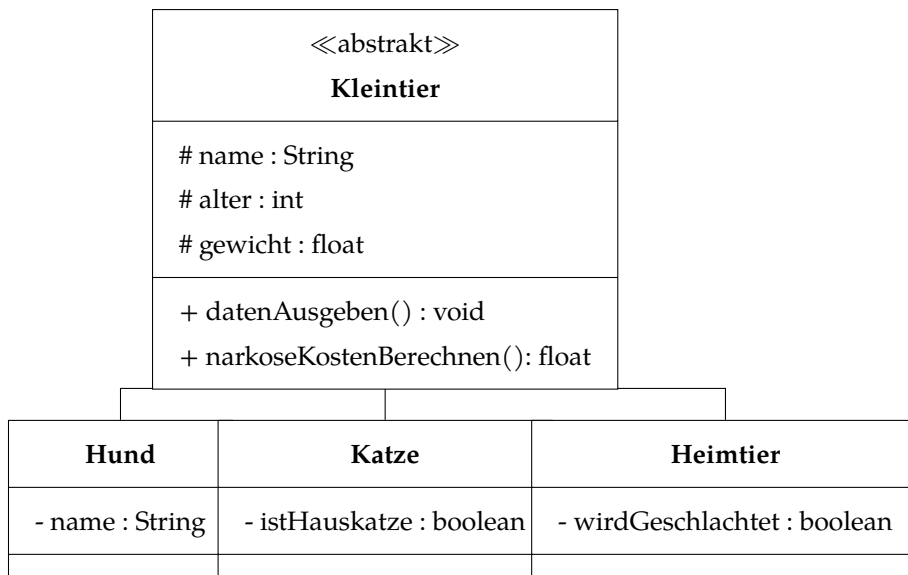
Einfügen in der abstrakten Klasse *Heimtier*. Fertiges Klassendiagramm siehe 1 (d)

```
public void datenAusgeben()
{
    System.out.println("Name: " + name + ", Alter: " + alter);
}
```

- (d) Die Kosten für eine Narkose setzen sich zusammen aus einer Grundgebühr, die von der Tierart abhängt, und aus den Kosten für das verwendete Narkotikum.



Diese wiederum berechnen sich aus dem Preis des Narkotikums pro kg Körpergewicht multipliziert mit dem Gewicht des Tieres. Ergänzen Sie das Klassendiagramm entsprechend um die Methode `narkosekostenBerechnen()`, die die Kosten für eine Narkose auf dem Bildschirm ausgibt, und implementieren Sie sie.



Einfügen in der abstrakten Klasse *Heimtier*.

```
public class Hund extends Kleintier {
    private boolean reinrassig;

    public Hund(String name, int alter, float gewicht, boolean reinrassig) {
        super(name, alter, gewicht);
        this.reinrassig = reinrassig;
        narkoseGrundGebuehr = 1.50f;
    }

    public void datenAusgeben() {
        System.out.println("Name: " + name + ", Alter: " + alter + " Jahre");

        if (reinrassig) {
            System.out.println("Der Hund ist reinrassig.");
        } else {
            System.out.println("Der Hund ist nicht reinrassig.");
        }
    }
}
```

Code-Beispiel auf Github ansehen: [src/main/java/org/bschlangaul/aufgaben/oomup/klassendiagramm/kleintierpraxis/Hund.java](https://github.com/bschlangaul/aufgaben/oomup/klassendiagramm/kleintierpraxis/Hund.java)

Einfügen in den Konstruktor der Klasse *Hund*:

```
narkoseGrundGebuehr = 1.50f;
```

Einfügen in den Konstruktor der Klasse *Katze*:

```
narkoseGrundGebuehr = 1f;
```

Einfügen in den Konstruktor der Klasse *Heimtier*:

```
narkoseGrundGebuehr = 2f;
```

- (e) Falls es sich bei dem zu behandelnden Tier um einen Hund handelt, soll die Methode `datenAusgeben()` (siehe Teilaufgabe c) dies zusammen mit dem Namen und dem Alter des Tieres auf dem Bildschirm ausgeben. Außerdem soll in diesem Fall ausgegeben werden, ob der Hund reinrassig ist oder nicht.

Lösungsvorschlag

Einfügen in der Klasse *Hund*:

```
public void datenAusgeben() {  
    System.out.println("Name: " + name + ", Alter: " + alter + " Jahre");  
  
    if (reinrassig) {  
        System.out.println("Der Hund ist reinrassig.");  
    } else {  
        System.out.println("Der Hund ist nicht reinrassig.");  
    }  
}
```

Code-Beispiel auf Github ansehen: [src/main/java/org/bschlangaul/aufgaben/oomup/klassendiagramm/kleintierpraxis/Hund.java](https://github.com/bschlangaul/aufgaben/oomup/klassendiagramm/kleintierpraxis/Hund.java)

## 66115 / 2021 / Frühjahr / Thema 2 / Aufgabe 4

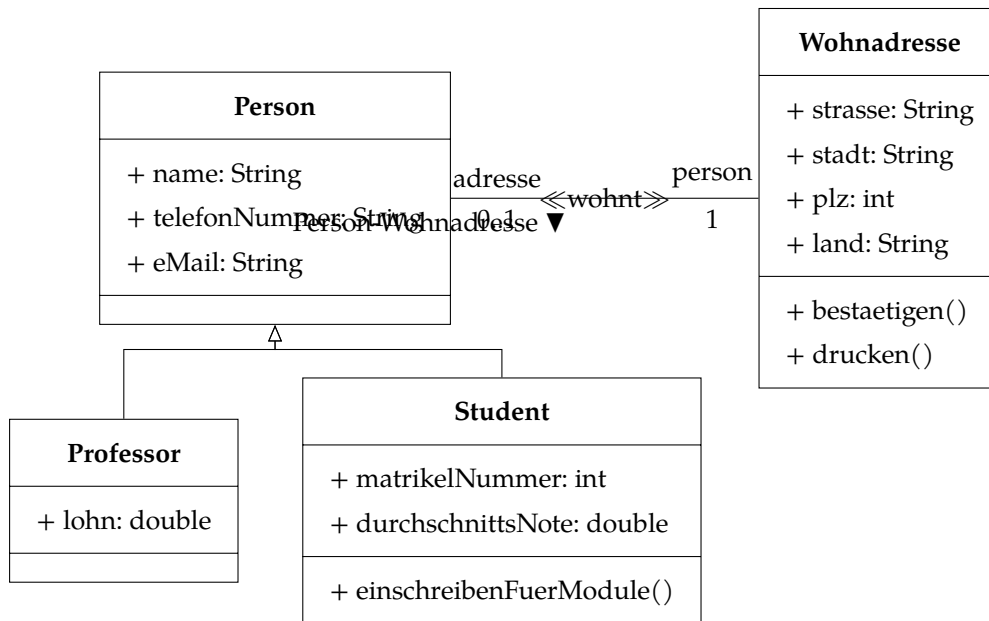
Gegeben ist der folgende Sachverhalt.

Jede **Person** hat einen *Namen*, eine *Telefonnummer* und *E-Mail*.

Jede **Wohnadresse** wird von nur einer Person bewohnt. Es kann aber sein, dass einige Wohnadressen nichtbewohnt sind. Den Wohnadressen sind je eine *Strasse*, eine *Stadt*, eine *PLZ* und ein *Land* zugeteilt. Alle Wohnadressen können bestätigt werden und als Beschriftung (für Postversand) gedruckt werden.

Es gibt zwei Sorten von **Personen**: **Student**, welcher sich für ein *Modul einschreiben* kann und **Professor**, welcher einen *Lohn* hat. Der Student besitzt eine *Matrikelnummer* und eine *Durchschnittsnote*.

Modellieren Sie diesen Sachverhalt mit einem UML Klassendiagramm.



## 66115 / 2021 / Frühjahr / Thema 2 / Aufgabe 4

Max und Sarah haben ein Ziel. Jeder Gast, der ihr Restaurant, das „fleißige Bienchen“ besucht, soll so schnell wie möglich satt werden. Um das zu erreichen arbeiten sie auf Hochtouren. Wenn ein Gast sein Essen bestellt, wird das sofort in der Küche gehört. Max fängt direkt an das Gericht zuzubereiten. Man hat es noch nicht bewiesen, aber wahrscheinlich ist er der schnellste Koch der Welt und dementsprechend dauert das Kochen nur wenige Minuten. Ist das Gericht fertig rennt Sarah zu den Gästen und serviert es. Manchmal verliert sie dabei, weil sie so schnell rennt, Teile des Essens. Dann muss sie zurück in die Küche, in der das Gericht neu zubereitet wird. Wenn aber alles gut gegangen ist, kann der Gast schnell essen. Gäste mit kleinem Hunger sind nach dem Essen satt. Hungrige Gäste bestellen noch einen Nachtisch der schon bereit steht, damit keine Zeit verloren geht. Ist dieser aufgegessen sind auch die hungrigsten Gäste satt. Sarah und Max haben herausgefunden, dass die Gäste beim Bezahlen der Rechnung viel mehr Trinkgeld geben, wenn sie innerhalb von einer halben Stunde satt sind. Aber egal, wie viel Geld sie bekommen. Max und Sarah sind glücklich über jeden Gast der sie besucht.

## 66115 / 2021 / Frühjahr / Thema 2 / Aufgabe 4

Ordnen Sie die gegebenen Diagrammart der Fragestellung zu, die das jeweilige Diagramm am besten beantwortet.

- (a) Wie sind die Daten und das Verhalten meines Systems im Detail strukturiert?

Lösungsvorschlag

Klassendiagramm

- (b) Wie sieht ein Schnappschuss meines Systems zur Ausführungszeit aus?

Lösungsvorschlag

Vererbung  
Generalisierung  
Spezialisierung

Objektdiagramm

- (c) Wie verhält sich das System in einem bestimmten Zustand bei gewissen Ereignissen?

Lösungsvorschlag

Zustandsdiagramm

- (d) Wie läuft die Kommunikation in meinem System ab?

Lösungsvorschlag

Sequenzdiagramm

- (e) Wie realisiert mein System ein bestimmtes Verhalten?

Lösungsvorschlag

Aktivitätsdiagramm

- (f) Was soll mein geplantes System leisten?

Lösungsvorschlag

Anwendungsfalldiagramm

- (g) Welche Teile einer komplexen Struktur arbeiten wie zusammen, um eine bestimmte Funktion zu erfüllen?

Lösungsvorschlag

Kommunikationsdiagramm

## 66115 / 2021 / Frühjahr / Thema 2 / Aufgabe 4

Eine kleine Bank bietet drei Arten von Konten an: Girokonten, Sparkonten und Geschäftskonten. Alle drei Kontoarten haben die Methoden `Einzahlen`, `Abheben` und `KontostandGeben` sowie die Attribute `kontostand` und `kontonummer`.

- Sparkonten haben einen Zinssatz und eine Methode `Verzinsen`, die den Jahreszins zum Guthaben addiert. Maximalbetrag beim Abheben ist der aktuelle Kontostand.
  - Girokonten können um bis 2000 € überzogen werden (Dispokredit).
  - Geschäftskonten haben einen variablen Dispokredit, der über die Methode `DispokreditSetzen` festgelegt wird; der Startwert für den Dispokredit wird mit dem Konstruktor beim Einrichten des Kontos als Parameter mitgegeben.
- (a) Überlege dir, welche Konten Generalisierungen bzw. Spezialisierungen anderer Konten sind. Warum ist es sinnvoll, eine Klasse Konto als oberste Klasse Generalisierungshierarchie einzuführen?

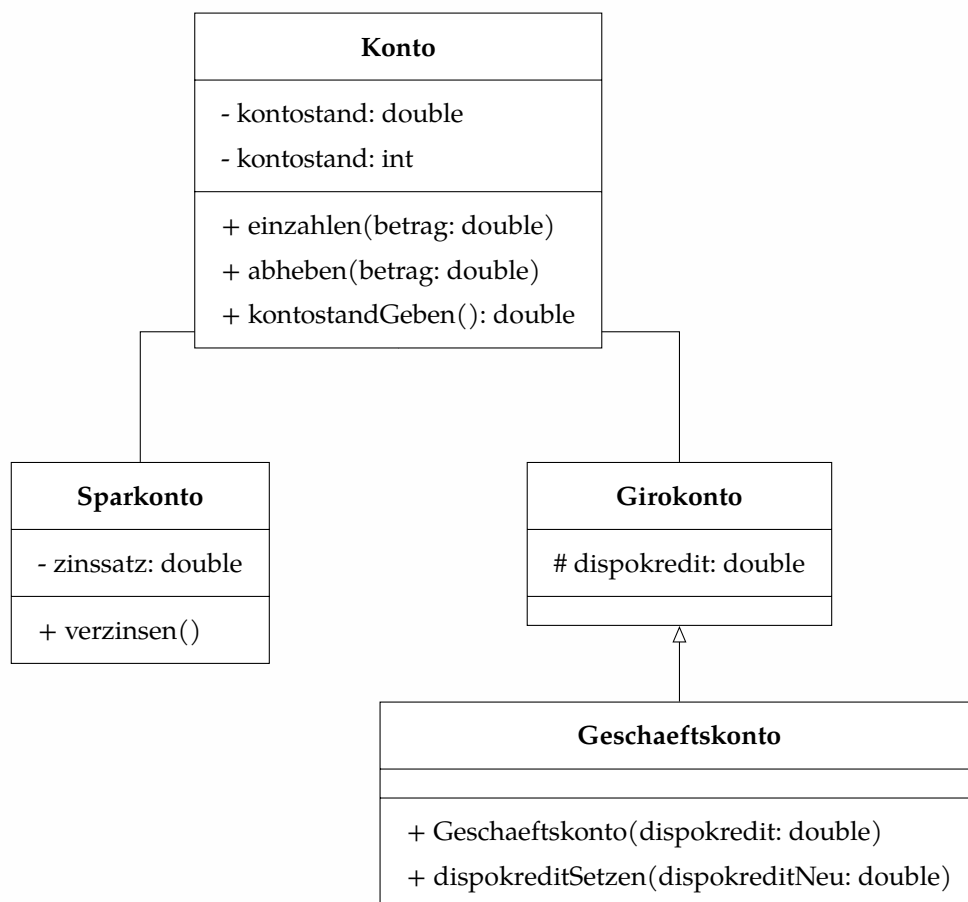
Da alle Konten die Methoden `einzahlen()`, `abheben()` und `kontostandGeben()` sowie die Attribute `kontostand` und `kontonummer` besitzen, bietet sich deren Verwaltung in einer einzigen Klasse an. Da jede Kontoart aber auch individuelle Eigenschaften bzw. Methoden hat, muss es für jede auch eine eigene Klasse geben. Daher bietet es sich an, die Gemeinsamkeiten in eine Oberklasse `Konto` auszulagern (Generalisierung).

- (b) Entwirf ein Klassendiagramm für die Klassen `Konto`, `Girokonto`, `Sparkonto`, `Geschaeftskonto`.

Da das Geschäftskonto genauso wie das Girokonto einen Dispokredit besitzt, dieser nur anders festgelegt wird, wurde bei der Modellierung die Klasse `Geschaeftskonto` als Unterklasse der Klasse `Girokonto` umgesetzt.

Die Oberklasse `Konto` wurde abstrakt modelliert, da von ihr direkt keine Objekte erzeugt werden können.

Die Attribute `kontostand` und `kontonummer` in der Klasse `Konto` haben den Sichtbarkeitsmodifikator `private`, da die Unterklassen nie direkt auf die Attribute zugreifen, sondern die zur Verfügung stehenden Methoden dafür verwenden.



- (c) Implementiere die Klassen in einem eigenen Projekt und teste die vorhandenen Methoden.

Implementierung in Java

```
public abstract class Konto {
    private double kontostand;
    @SuppressWarnings("unused")
    private int kontonummer;

    public Konto(int kontonummer) {
        this.kontonummer = kontonummer;
        kontostand = 0;
    }

    public void einzahlen(double betrag) {
        kontostand = kontostand + betrag;
    }

    public void abheben(double betrag) {
        // Ob abgehoben werden darf, entscheidet die Methode der jeweiligen
        ↪ Unterklasse.
        kontostand = kontostand - betrag;
    }

    public double kontostandGeben() {
        return kontostand;
    }
}
```

Code-Beispiel auf Github ansehen: [src/main/java/org/bschlangaul/aufgaben/oomup/pu\\_3/bankverwaltung/Konto.java](https://github.com/bschlangaul/aufgaben/oomup/pu_3/bankverwaltung/Konto.java)

```
public class Sparkonto extends Konto {
    private double zinssatz;

    public Sparkonto(int kontonummer, double zinssatz) {
        super(kontonummer);
        // Aufruf des Konstruktors der Oberklasse
        this.zinssatz = zinssatz;
    }

    /**
     * Überschreiben der Methode abheben() aus der Oberklasse. Ist genügend
     ↪ Geld auf
     * dem Sparkonto...? dann darf man den gewünschten Betrag abheben, sonst
     ↪ nicht.
     */
    public void abheben(double betrag) {
        if (kontostandGeben() >= betrag) {
            super.abheben(betrag);
        }
    }

    /**
     * Der aktuelle Kontostand wird mit dem Zinssatz verrechnet und der sich
     ↪ daraus
     */
}
```



```
    * ergebende Betrag (= Zinsen) dem Konto gutgeschrieben.
    */
    public void verzinsen() {
        einzahlen(kontostandGeben() * zinssatz);
    }
}
```

Code-Beispiel auf Github ansehen: [src/main/java/org/bschlangaul/aufgaben/oomup/pu\\_3/bankverwaltung/Sparkonto.java](https://github.com/orgs/bschlangaul/repositories?q=oomup/pu_3/bankverwaltung/Sparkonto.java)

```
public class Girokonto extends Konto {
    /**
     * Die Unterklasse muss auch auf das Attribut zugreifen können.
     */
    protected double dispokredit;

    public Girokonto(int kontonummer) {
        super(kontonummer);
        dispokredit = 2000;
    }

    /**
     * Die Methode abheben() kann direkt von der Oberklasse GIROKONTO und die
     * Methoden einzahlen() und kontostandGeben() von der Oberklasse KONTO
    → genutzt
     * werden.
     *
     * Überschreiben der Methode abheben() der Klasse KONTO
     * Ist genügend Geld auf dem Konto bzw. reicht der Dispokredit aus...
     * ...dann darf man den gewünschten Betrag abheben, sonst nicht.
     */
    public void abheben(double betrag) {
        if (kontostandGeben() + dispokredit >= betrag) {
            super.abheben(betrag);
        }
    }
}
```

Code-Beispiel auf Github ansehen: [src/main/java/org/bschlangaul/aufgaben/oomup/pu\\_3/bankverwaltung/Girokonto.java](https://github.com/orgs/bschlangaul/repositories?q=oomup/pu_3/bankverwaltung/Girokonto.java)

```
public class Geschaeftskonto extends Girokonto {

    /**
     * Für das Geschäftskonto wird der Dispo individuell festgelegt.
     *
     * @param kontonummer Die Kontonummer
     * @param dispo Der maximale Rahme des Dispokredits.
     */
    public Geschaeftskonto(int kontonummer, double dispo) {
        super(kontonummer);
        dispokredit = dispo;
    }

    public void dispokreditSetzen(double dispokreditNeu) {
```

```

        dispokredit = dispokreditNeu;
    }
}

```

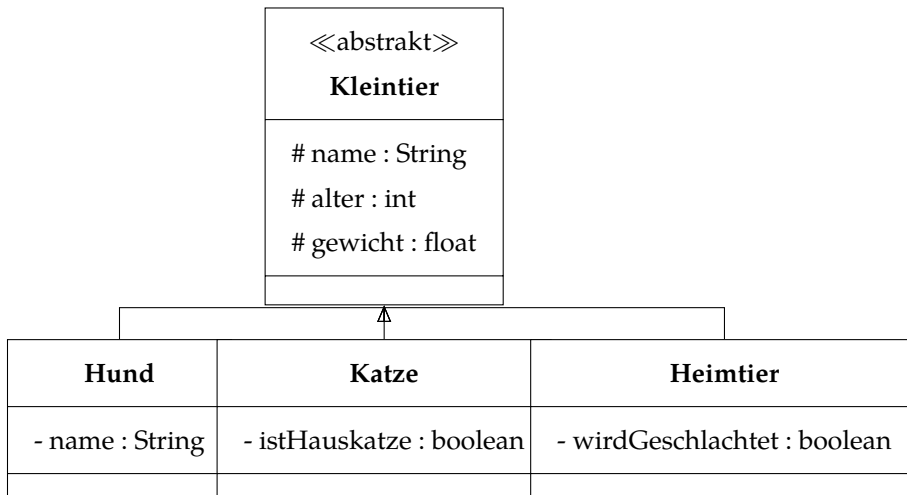
Vererbung  
Klassendiagramm  
Implementierung in Java

Code-Beispiel auf Github ansehen: [src/main/java/org/bschlangaul/aufgaben/oomup/pu\\_3/bankverwaltung/Geschaeftskonto.java](https://src/main/java/org/bschlangaul/aufgaben/oomup/pu_3/bankverwaltung/Geschaeftskonto.java)

## 66115 / 2021 / Frühjahr / Thema 2 / Aufgabe 4

In der Kleintierpraxis Dr. Huf werden *Hunde*, *Katzen* und *Heimtiere*<sup>2</sup> behandelt. In der Praxissoftware werden von jedem Tier *Name*, *Alter* (in Jahren) und *Gewicht* (in kg mit mindestens 2 Dezimalen) erfasst. Bei einem *Hund* wird zusätzlich aufgenommen, ob er *reinrassig* ist, bei einer *Katze*, ob sie ausschließlich *in der Wohnung gehalten* wird, und beim *Heimtier*, ob es zur *Lebensmittellieferung* dient.

- (a) Erstellen Sie ein entsprechendes Klassendiagramm (zunächst ohne Methoden), das den oben beschriebenen Sachverhalt geeignet erfasst.



- (b) Implementieren Sie die Klassen gemäß des Modells aus Teilaufgabe a in der Programmierungsumgebung BlueJ. Beachten Sie dabei die abstrakte Oberklasse!

### Klasse *Kleintier*

```

public abstract class Kleintier {
    protected String name;
    protected int alter;
    protected float gewicht;
}

```

Code-Beispiel auf Github ansehen: [src/main/java/org/bschlangaul/aufgaben/oomup/klassendiagramm/kleintierpraxis/Kleintier.java](https://src/main/java/org/bschlangaul/aufgaben/oomup/klassendiagramm/kleintierpraxis/Kleintier.java)

<sup>2</sup>Unter Heimtiere versteht man hier Kleintiere (d. h. keine Rinder, Pferde etc.), die zu Hause gehalten werden und keine Hunde oder Katzen sind (beispielsweise Meerschweinchen oder Kaninchen). Heimtiere können auch Nutztiere sein, die zur Schlachtung gehalten werden (z. B. Hühner). In diesem Fall muss ein Tierarzt darauf achten, nur Medikamente anzuwenden, die für Lebensmittel liefernde Tiere zugelassen sind.

```
public Kleintier(String name, int alter, float gewicht) {  
    this.name = name;  
    this.alter = alter;  
    this.gewicht = gewicht;  
}
```

Code-Beispiel auf Github ansehen: [src/main/java/org/bschlangaul/aufgaben/oomup/klassendiagramm/kleintierpraxis/Kleintier.java](#)

## Klasse *Hund*

```
public class Hund extends Kleintier {  
    private boolean reinrassig;  
  
    public Hund(String name, int alter, float gewicht, boolean reinrassig) {  
        super(name, alter, gewicht);  
        this.reinrassig = reinrassig;  
    }  
}
```

Code-Beispiel auf Github ansehen: [src/main/java/org/bschlangaul/aufgaben/oomup/klassendiagramm/kleintierpraxis/Hund.java](#)

## Klasse *Katze*

```
public class Katze extends Kleintier {  
    @SuppressWarnings("unused")  
    private boolean istHauskatze;  
  
    public Katze(String name, int alter, float gewicht, boolean istHauskatze) {  
        super(name, alter, gewicht);  
        this.istHauskatze = istHauskatze;  
    }  
}
```

Code-Beispiel auf Github ansehen: [src/main/java/org/bschlangaul/aufgaben/oomup/klassendiagramm/kleintierpraxis/Katze.java](#)

## Klasse *Heimtier*

```
public class Heimtier extends Kleintier {  
    @SuppressWarnings("unused")  
    private boolean wirdGeschlachtet;  
  
    public Heimtier(String name, int alter, float gewicht, boolean  
↳ wirdGeschlachtet) {  
        super(name, alter, gewicht);  
        this.wirdGeschlachtet = wirdGeschlachtet;  
    }  
}
```

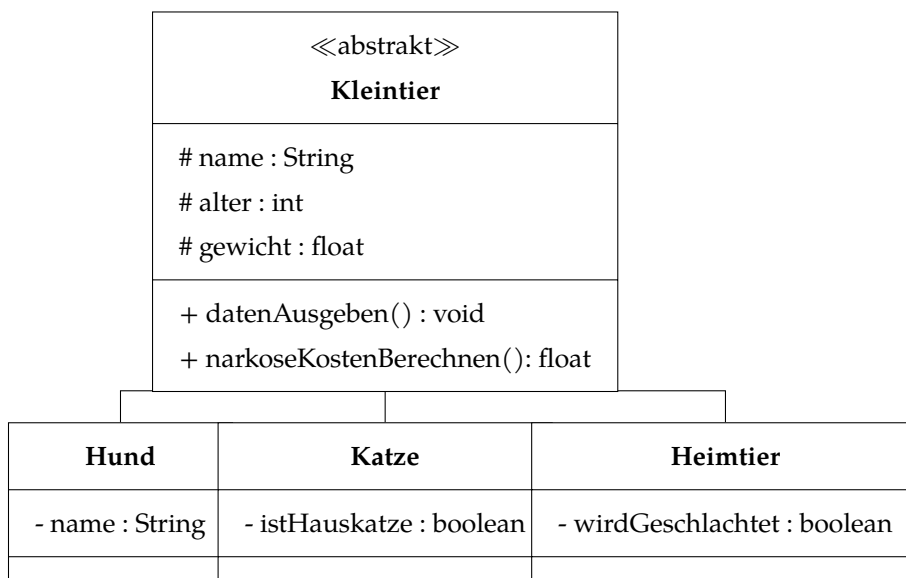
Code-Beispiel auf Github ansehen: [src/main/java/org/bschlangaul/aufgaben/oomup/klassendiagramm/kleintierpraxis/Heimtier.java](#)

- (c) Die Praxissoftware verfügt über eine Methode `datenAusgeben()`, die den Namen und das Alter eines Tieres auf dem Bildschirm ausgibt. Fügen Sie im Klassendiagramm die Methode an geeigneter Stelle ein und implementieren Sie sie.

Einfügen in der abstrakten Klasse *Heimtier*. Fertiges Klassendiagramm siehe 1 (d)

```
public void datenAusgeben()
{
    System.out.println("Name: " + name + ", Alter: " + alter);
}
```

- (d) Die Kosten für eine Narkose setzen sich zusammen aus einer Grundgebühr, die von der Tierart abhängt, und aus den Kosten für das verwendete Narkotikum. Diese wiederum berechnen sich aus dem Preis des Narkotikums pro kg Körpergewicht multipliziert mit dem Gewicht des Tieres. Ergänzen Sie das Klassendiagramm entsprechend um die Methode `narkosekostenBerechnen()`, die die Kosten für eine Narkose auf dem Bildschirm ausgibt, und implementieren Sie sie.



Einfügen in der abstrakten Klasse *Heimtier*.

```
public class Hund extends Kleintier {
    private boolean reinrassig;

    public Hund(String name, int alter, float gewicht, boolean reinrassig) {
        super(name, alter, gewicht);
        this.reinrassig = reinrassig;
        narkoseGrundGebuehr = 1.50f;
    }

    public void datenAusgeben() {
        System.out.println("Name: " + name + ", Alter: " + alter + " Jahre");

        if (reinrassig) {
            System.out.println("Der Hund ist reinrassig.");
        } else {

```

```

        System.out.println("Der Hund ist nicht reinrassig.");
    }
}

```

Code-Beispiel auf Github ansehen: [src/main/java/org/bschlangaul/aufgaben/oomup/klassendiagramm/kleintierpraxis/Hund.java](https://github.com/bschlangaul/aufgaben/oomup/klassendiagramm/kleintierpraxis/Hund.java)

Einfügen in den Konstruktor der Klasse *Hund*:

```
narkoseGrundGebuehr = 1.50f;
```

Einfügen in den Konstruktor der Klasse *Katze*:

```
narkoseGrundGebuehr = 1f;
```

Einfügen in den Konstruktor der Klasse *Heimtier*:

```
narkoseGrundGebuehr = 2f;
```

- (e) Falls es sich bei dem zu behandelnden Tier um einen Hund handelt, soll die Methode `datenAusgeben()` (siehe Teilaufgabe c) dies zusammen mit dem Namen und dem Alter des Tieres auf dem Bildschirm ausgeben. Außerdem soll in diesem Fall ausgegeben werden, ob der Hund reinrassig ist oder nicht.

Lösungsvorschlag

Einfügen in der Klasse *Hund*:

```

public void datenAusgeben() {
    System.out.println("Name: " + name + ", Alter: " + alter + " Jahre");

    if (reinrassig) {
        System.out.println("Der Hund ist reinrassig.");
    } else {
        System.out.println("Der Hund ist nicht reinrassig.");
    }
}

```

Code-Beispiel auf Github ansehen: [src/main/java/org/bschlangaul/aufgaben/oomup/klassendiagramm/kleintierpraxis/Hund.java](https://github.com/bschlangaul/aufgaben/oomup/klassendiagramm/kleintierpraxis/Hund.java)

## 66115 / 2021 / Frühjahr / Thema 2 / Aufgabe 4

Gegeben sei folgende rekursive Methodendeklaration in der Sprache Java. Es wird als Vorbedingung vorausgesetzt, dass die Methode `sum` nur für Werte  $n \geq 0$  aufgerufen wird.

```

public static int sum(int n) {
    if (n <= 0) {
        return 0;
    }
    return n + sum(n - 1);
}

```

Beweisen Sie mittels vollständiger Induktion, dass der Methodenaufruf `sum(n)` die Summe der ersten  $n$  aufeinanderfolgenden natürlichen Zahlen für alle  $n \geq 0$  berechnet, wobei gilt

$$\sum_{k=0}^n k = \frac{n(n+1)}{2}$$

Lösungsvorschlag

**Induktionsanfang**

— Beweise, dass  $A(1)$  eine wahre Aussage ist. \_\_\_\_\_

$$\sum_{k=0}^0 k = \frac{0(0+1)}{2} = \frac{0}{2} = 0$$

$$\text{sum}(0) = 0$$

**Induktionsvoraussetzung**

— Die Aussage  $A(k)$  ist wahr für ein beliebiges  $k \in \mathbb{N}$ . \_\_\_\_\_

$$\sum_{k=0}^n k = \frac{n(n+1)}{2}$$

$$\text{sum}(n) = n + \frac{(n-1)((n-1)+1)}{2} = n + \frac{(n-1)n}{2}$$

**Induktionsschritt**

— Beweise, dass wenn  $A(n=k)$  wahr ist, auch  $A(n=k+1)$  wahr sein muss. \_\_\_\_\_

$$\sum_{k=0}^{n+1} k = \frac{(n+1)((n+1)+1)}{2}$$

$$\begin{aligned}
\text{sum}(n+1) &= (n+1) + \frac{((n+1)-1)(n+1)}{2} && n+1-1=n \\
&= (n+1) + \frac{n(n+1)}{2} && (n+1) \text{ eingesetzt} \\
&= \frac{2(n+1)}{2} + \frac{n(n+1)}{2} && (n+1) \text{ als Bruch geschrieben} \\
&= \frac{2(n+1) + n(n+1)}{2} && \text{Hauptnenner 2} \\
&= \frac{(2+n)(n+1)}{2} && (n+1) \text{ ausgeklammert} \\
&= \frac{(n+2)(n+1)}{2} && \text{Kommutativgesetz angewandt} \\
&= \frac{(n+1)(n+2)}{2} && \text{getauscht nach Kommutativgesetz} \\
&= \frac{(n+1)((n+1)+1)}{2} && \text{mit } (n+1) \text{ an der Stelle von } n
\end{aligned}$$

```

import static org.junit.Assert.assertEquals;

import org.junit.Test;

public class GaussTest {

    private void teste(int n, int erwartet) {
        assertEquals(Gauss.sum(n), erwartet);
    }

    @Test
    public void teste() {
        teste(0, 0);
        teste(1, 1);
        teste(2, 3);
        teste(3, 6);
        teste(4, 10);
        teste(5, 15);
        teste(6, 21);
        teste(7, 28);
        teste(8, 36);
        teste(9, 45);
        teste(10, 55);
        teste(11, 66);
    }
}

```

Code-Beispiel auf Github ansehen: [src/test/java/org/bschlangaul/aufgaben/aud/induktion/GaussTest.java](https://github.com/bschlangaul/aufgaben/aud/induktion/GaussTest.java)

## 66115 / 2021 / Frühjahr / Thema 2 / Aufgabe 4

Gegeben sei folgende Methode:

```
public class GeoSum {
    // Math.pow(q, n) == q^n
    double geoSum(int n, double q) {
        if (n == 0) {
            return 1 - q;
        } else {
            return (1 - q) * Math.pow(q, n) + geoSum(n - 1, q);
        }
    }
}
```

Code-Beispiel auf Github ansehen: [src/main/java/org/beschlangaul/aufgaben/sosy/totale\\_korrektheit/GeoSum.java](https://github.com/src/main/java/org/beschlangaul/aufgaben/sosy/totale_korrektheit/GeoSum.java)

Weisen Sie mittels vollständiger Induktion nach, dass

$$\text{geoSum}(n, q) = 1 - q^{n+1}$$

Dabei können Sie davon ausgehen, dass  $q > 0, n \in \mathbb{N}_0$

Lösungsvorschlag

### Induktionsanfang

— Beweise, dass  $A(1)$  eine wahre Aussage ist. \_\_\_\_\_

$$f(0) : \text{geoSum}(0, q) = 1 - q^{0+1} = 1 - q^1 = 1 - q$$

### Induktionsvoraussetzung

— Die Aussage  $A(k)$  ist wahr für ein beliebiges  $k \in \mathbb{N}$ . \_\_\_\_\_

$$f(n) : \text{geoSum}(n, q) = 1 - q^{n+1}$$

### Induktionsschritt

— Beweise, dass wenn  $A(n = k)$  wahr ist, auch  $A(n = k + 1)$  wahr sein muss. \_\_\_\_\_

$$\begin{aligned} f(n) &= \text{geoSum}(n, q) \\ &= (1 - q) \cdot q^n + \text{geoSum}(n - 1, q) \\ &= (1 - q) \cdot q^n + 1 - q^{(n-1)+1} \\ &= (1 - q) \cdot q^n + 1 - q^n \end{aligned}$$

Java-Code in Mathe-Formel umgewandelt

für rekursiven Methodenaufwurf gegebene Formel eingesetzt

Addition im Exponent



$$\begin{aligned}
f(n+1) &= \text{geoSum}(n+1, q) \\
&= (1-q) \cdot q^{n+1} + 1 - q^{n+1} && \text{von Java konvertierte Formel verwendet und } n+1 \text{ eingesetzt} \\
&= q^{n+1} - q^{(n+1)+1} + 1 - q^{n+1} && \text{ausmultipliziert} \\
&= -q^{(n+1)+1} + 1 && q^{n+1} - q^{n+1} = 0 \\
&= 1 - q^{(n+1)+1} && \text{Kommutativgesetz der Addition} \\
&= 1 - q^{(n+1)+1} && \text{was zu zeigen war}
\end{aligned}$$

## 66115 / 2021 / Frühjahr / Thema 2 / Aufgabe 4

Die schrittweise Berechnung der Summe der ersten  $n$  ungeraden Zahlen legt die Vermutung nahe: Die Summe aller ungeraden Zahlen von 1 bis  $2n - 1$  ist gleich dem Quadrat von  $n$ :

$$\begin{aligned}
1 &= 1 = 1^2 \\
1 + 3 &= 4 = 2^2 \\
1 + 3 + 5 &= 9 = 3^2 \\
1 + 3 + 5 + 7 &= 16 = 4^2
\end{aligned}$$

Folgende Java-Methode berechnet die Summer aller ungeraden Zahlen:

```

public static int oddSum(int n) {
    if (n <= 1) {
        return 1;
    }
    return 2 * n - 1 + oddSum(n - 1);
}

```

Code-Beispiel auf Github ansehen: [src/main/java/org/bschlangaul/aufgaben/aud/induktion/Maurolicus.java](https://github.com/bschlangaul/aufgaben/aud/induktion/Maurolicus.java)

Beweisen Sie mittels vollständiger Induktion, dass der Methodenaufruf `oddSum(n)` die Summe aller ungeraden Zahlen von 1 bis nur  $n$ -ten ungeraden Zahl berechnet, wobei gilt:

$$\sum_{i=1}^n (2i - 1) = n^2$$

Lösungsvorschlag

### Induktionsanfang

— Beweise, dass  $A(1)$  eine wahre Aussage ist. \_\_\_\_\_

$$\sum_{i=1}^1 (2i - 1) = 2 \cdot 1 - 1 = 1 = 1^2$$

$$\text{oddSum}(1) = 1 = 1^2$$

**Induktionsvoraussetzung**

— Die Aussage  $A(k)$  ist wahr für ein beliebiges  $k \in \mathbb{N}$ . —

$$\sum_{i=1}^n (2i - 1) = n^2$$

$$\text{oddSum}(n) = 2n - 1 + (n - 1)^2$$

**Induktionsschritt**

— Beweise, dass wenn  $A(n = k)$  wahr ist, auch  $A(n = k + 1)$  wahr sein muss. —

$$\begin{aligned} \text{oddSum}(n) &= 2(n + 1) - 1 + ((n + 1) - 1)^2 \\ &= 2(n + 1) - 1 + n^2 \\ &= 2n + 2 + n^2 - 1 && \text{ausmultiplizieren} \\ &= 2n + 1 + n^2 && 2 - 1 = 1 \\ &= n^2 + 2n + 1 && \text{Kommutativgesetz} \\ &= (n + 1)^2 && \text{mit erster Binomischer Formel: } (a + b)^2 = a^2 + 2ab + b^2 \end{aligned}$$

```
import static org.junit.Assert.assertEquals;

import org.junit.Test;

public class MaurolicusTest {

    private void teste(int n, int erwartet) {
        assertEquals(Maurolicus.oddSum(n), erwartet);
    }

    @Test
    public void teste() {
        teste(1, 1);
        teste(2, 4);
        teste(3, 9);
        teste(4, 16);
        teste(5, 25);
        teste(6, 36);
        teste(7, 49);
        teste(8, 64);
        teste(9, 81);
        teste(10, 100);
        teste(11, 121);
    }
}
```

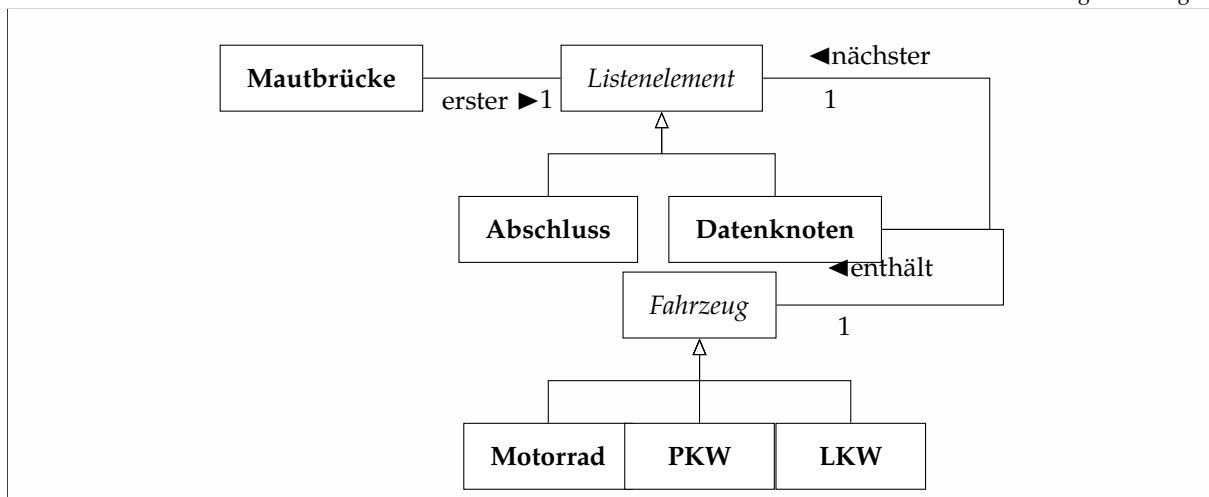
}

Code-Beispiel auf Github ansehen: [src/test/java/org/bschlangaul/aufgaben/aud/induktion/MaurolicusTest.java](https://github.com/bschlangaul/aufgaben/aud/induktion/MaurolicusTest.java)Einfach-verkettete Liste  
Klassendiagramm  
Kompositum (Composite)  
Einfach-verkettete Liste  
Kompositum (Composite)

## 66115 / 2021 / Frühjahr / Thema 2 / Aufgabe 4

Für die Umsetzung der Maut auf deutschen Autobahnen soll eine Java-basierte Lösung entworfen werden. Dazu sollen alle Fahrzeuge, die von einer Mautbrücke erfasst werden, in einer *einfach verketteten Liste* abgelegt werden. Um einen besseren Überblick über die Einnahmen zu erhalten, soll zwischen *LKWs*, *PKWs* und *Motorrädern* unterschieden werden. Als Informatiker schlagen Sie eine *heterogene Liste* zur Realisierung vor. Notieren Sie unter Verwendung des *Entwurfsmusters Kompositum* ein entsprechendes *Klassendiagramm* zur Realisierung der Lösung für eine Mautbrücke. Auf die Angabe von Attributen und Methoden kann verzichtet werden. Kennzeichnen Sie in Ihrem Klassendiagramm die *abstrakten Klassen* und benennen Sie die bestehenden *Beziehungen*.

Lösungsvorschlag



## 66115 / 2021 / Frühjahr / Thema 2 / Aufgabe 4

Erstellen Sie ein Deutsch-Englisch Wörterbuch. Verwenden Sie dazu eine einfach verkettete Liste mit Kompositum. Identifizieren Sie die benötigten Klassen, legen Sie das Wörterbuch an und implementieren Sie anschließend die geforderten Methoden.

- Ein Listenelement, welches immer jeweils auf seinen Nachfolger verweisen kann, enthält jeweils einen Eintrag des Wörterbuchs. Ein Eintrag besteht aus dem deutschen und dem zugehörigen englischen Wort. Diese können natürlich jeweils zurückgegeben werden.
- Mit der Methode `einfuegen (String deutsch, String englisch)` soll ein neuer Eintrag in das Wörterbuch eingefügt werden können. Wie in jedem Wörterbuch müssen die (deutschen) Einträge jedoch alphabetisch sortiert sein, sodass nicht an einer beliebigen Stelle eingefügt werden kann. Um die korrekte Einfügeposition zu finden, ist das Vergleichen von Strings notwendig. Recherchieren Sie dazu, wie die Methode `compareTo()` in Java funktioniert!

- Der Aufruf der Methode `uebersetze(String deutsch)` auf der Liste soll nun für ein übergebenes deutsches Wort die englische Übersetzung ausgeben.

Lösungsvorschlag

### Klasse WörterbuchEintrag

Die abstrakte Klasse im Kompositumentwurfsmuster von der sowohl die primitive Klasse als auch die Behälterklasse erben.

```
public abstract class WoerterbuchEintrag {  
    protected WortPaar nächstes;  
  
    protected WortPaar gibNächstes () {  
        return nächstes;  
    }  
  
    protected void setzeNächstes (WortPaar wortPaar) {  
        nächstes = wortPaar;  
    }  
}
```

Code-Beispiel auf Github ansehen: [src/main/java/org/bschlangaul/aufgaben/aud/listen/woerterbuch/WoerterbuchEintrag.java](https://github.com/bschlangaul/aufgaben/aud/listen/woerterbuch/WoerterbuchEintrag.java)

### Klasse WortPaar

Das Listenelement (die primitive Klasse im Kompositumentwurfsmuster).

```
public class WortPaar extends WoerterbuchEintrag {  
    private final String deutsch;  
  
    private final String englisch;  
  
    public WortPaar(String deutsch, String englisch) {  
        this.deutsch = deutsch;  
        this.englisch = englisch;  
    }  
  
    public String gibDeutschesWort() {  
        return deutsch;  
    }  
  
    public String gibEnglischesWort() {  
        return englisch;  
    }  
}
```

Code-Beispiel auf Github ansehen: [src/main/java/org/bschlangaul/aufgaben/aud/listen/woerterbuch/WortPaar.java](https://github.com/bschlangaul/aufgaben/aud/listen/woerterbuch/WortPaar.java)

### Klasse Wörterbuch

Die Behälterklasse im Kompositumentwurfsmuster.

```

*/
public class Woerterbuch extends WoerterbuchEintrag {

    public void einfügen(String deutsch, String englisch) {
        WortPaar wort = new WortPaar(deutsch, englisch);

        // Spezialbehandlung, wenn vor das erste Wortpaar des Wörterbuchs eingefügt
        // werden muss.
        WortPaar kopf = gibNächstes();
        if (kopf == null || kopf.gibDeutschesWort().compareTo(wort.gibDeutschesWort())
→ >= 0) {
            wort.setzeNächstes(kopf);
            setzeNächstes(wort);
            return;
        }
        WortPaar vergleichsWort = gibNächstes();
        while (vergleichsWort.gibNächstes() != null
            &&
→ vergleichsWort.gibNächstes().gibDeutschesWort().compareTo(wort.gibDeutschesWort())
→ < 0) {
            vergleichsWort = vergleichsWort.gibNächstes();
        }
        wort.setzeNächstes(vergleichsWort.gibNächstes());
        vergleichsWort.setzeNächstes(wort);
    }

    public String übersetze(String deutsch) {
        if (gibNächstes() == null) {
            return "Noch keine Wörter im Wörterbuch.";
        }
        WortPaar wort = gibNächstes();
        while (wort != null) {
            if (wort.gibDeutschesWort().equals(deutsch)) {
                return wort.gibEnglischesWort();
            }
            wort = wort.gibNächstes();
        }
        return "Es konnte keine passende Übersetzung gefunden werden";
    }
}

```

Code-Beispiel auf Github ansehen: [src/main/java/org/bschlangaul/aufgaben/aud/listen/woerterbuch/Woerterbuch.java](https://github.com/bschlangaul/aufgaben/aud/listen/woerterbuch/Woerterbuch.java)

## Test

```

import static org.junit.Assert.assertEquals;

import org.junit.Test;

public class WoerterbuchTest {

    @Test
    public void methodeÜbersetze() {
        Woerterbuch wörterbuch = new Woerterbuch();
    }
}

```

```
    assertEquals("Noch keine Wörter im Wörterbuch.",
→   wörterbuch.übersetze("Wassermelone"));
}

@Test
public void methodeEinfügen() {
    Woerterbuch wörterbuch = new Woerterbuch();
    wörterbuch.einfügen("Wassermelone", "Watermelon");
    assertEquals("Watermelon", wörterbuch.übersetze("Wassermelone"));
}

@Test
public void sortierung() {
    Woerterbuch wörterbuch = new Woerterbuch();
    wörterbuch.einfügen("Wassermelone", "Watermelon");
    wörterbuch.einfügen("Apfel", "Apple");
    wörterbuch.einfügen("Zitrone", "Lemon");
    wörterbuch.einfügen("Birne", "Pear");
    wörterbuch.einfügen("Klementine", "Clementine");

    WortPaar paar;
    paar = wörterbuch.gibNächstes();
    assertEquals("Apfel", paar.gibDeutschesWort());

    paar = paar.gibNächstes();
    assertEquals("Birne", paar.gibDeutschesWort());

    paar = paar.gibNächstes();
    assertEquals("Klementine", paar.gibDeutschesWort());

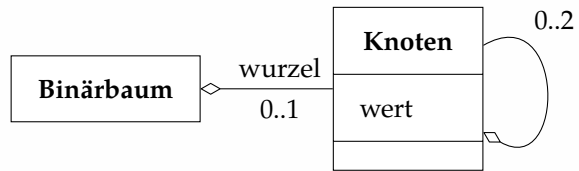
    paar = paar.gibNächstes();
    assertEquals("Wassermelone", paar.gibDeutschesWort());

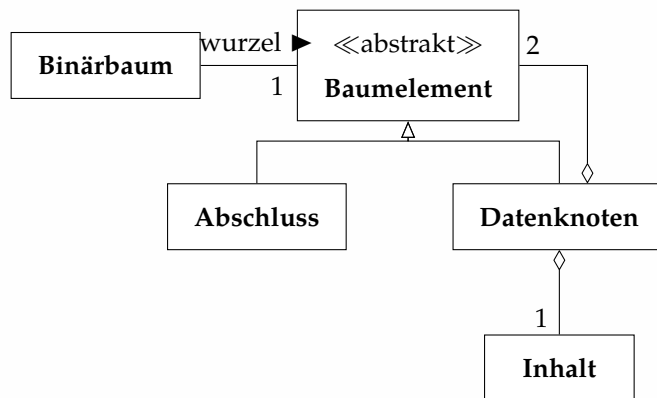
    paar = paar.gibNächstes();
    assertEquals("Zitrone", paar.gibDeutschesWort());
}
}
```

Code-Beispiel auf Github ansehen: [src/test/java/org/bschlangaul/aufgaben/aud/listen/woerterbuch/WoerterbuchTest.java](https://github.com/bschlangaul/aufgaben/aud/listen/woerterbuch/WoerterbuchTest.java)

## 66115 / 2021 / Frühjahr / Thema 2 / Aufgabe 4

- (a) Erstellen Sie ein Klassendiagramm für einen Binärbaum.

**einfacher Binärbaum**

**Binärbaum mit Kompositum**

- (b) Entwerfen Sie eine mögliche Implementierung zur Erzeugung eines binären Baumes in Java.

Lösungsvorschlag

**einfacher Binärbaum**

```

public class Binaerbaum {
    public Knoten wurzel;

    public void fügeEin(int wert) {
        if (wurzel == null) {
            wurzel = new Knoten(wert);
        } else {
            if (wert <= wurzel.gibWert()) {
                if (wurzel.gibLinks() != null) {
                    wurzel.gibLinks().fügeEin(wert);
                } else {
                    wurzel.setzeLinks(new Knoten(wert));
                }
            } else {
                if (wurzel.gibRechts() != null) {
                    wurzel.gibRechts().fügeEin(wert);
                } else {
                    wurzel.setzeRechts(new Knoten(wert));
                }
            }
        }
    }
}

```

Code-Beispiel auf Github ansehen: [src/main/java/org/bschlangaul/aufgaben/aud/baum/einfach/Binaerbaum.java](https://github.com/bschlangaul/aufgaben/aud/baum/einfach/Binaerbaum.java)

```

public class Knoten {

    private Knoten links;
    private Knoten rechts;
    private int wert;
}

```



```
public Knoten(int wert) {
    this.wert = wert;
}

public void setzeWert(int w) {
    wert = w;
}

public int gibWert() {
    return wert;
}

public void setzeLinks(Knoten l) {
    links = l;
}

public void setzeRechts(Knoten r) {
    rechts = r;
}

public Knoten gibLinks() {
    return links;
}

public Knoten gibRechts() {
    return rechts;
}

public void fügeEin(int wert) {
    if (wert <= this.gibWert()) {
        if (this.gibLinks() != null) {
            this.gibLinks().fügeEin(wert);
        } else {
            this.setzeLinks(new Knoten(wert));
        }
    } else {
        if (this.gibRechts() != null) {
            this.gibRechts().fügeEin(wert);
        } else {
            this.setzeRechts(new Knoten(wert));
        }
    }
}
}
```

Code-Beispiel auf Github ansehen: [src/main/java/org/bschlangaul/aufgaben/aud/baum/einfach/Knoten.java](https://github.com/bschlangaul/aufgaben/blob/master/aud/baum/einfach/Knoten.java)

## Binärbaum mit Kompositum

```
class Binaerbaum {
    private Bauelement wurzel;

    public Binaerbaum() {
        wurzel = new Abschluss();
    }

    public void setzeWurzel(Bauelement wurzel) {
        this.wurzel = wurzel;
    }

    public Bauelement gibWurzel() {
        return wurzel;
    }

    public int gibAnzahl() {
        return wurzel.gibAnzahl();
    }

    public void gibAus() {
        wurzel.gibAus();
    }
}
```

Code-Beispiel auf Github ansehen: [src/main/java/org/bschlangaul/aufgaben/aud/baum/kompositum/Binaerbaum.java](#)

```
abstract class Bauelement {
    public abstract void setztleLinks(Bauelement nl);

    public abstract void setzeRechts(Bauelement nr);

    public abstract Bauelement gibLinks();

    public abstract Bauelement gibRechts();

    public abstract Datenelement gibInhalt();

    public abstract int gibAnzahl();

    public abstract void gibAus();
}
```

Code-Beispiel auf Github ansehen: [src/main/java/org/bschlangaul/aufgaben/aud/baum/kompositum/Bauelement.java](#)

```
class Abschluss extends Bauelement {

    public void setztleLinks(Bauelement links) {
        System.out.println("Ein Abschluss hat kein linkes Element!");
    }

    public void setzeRechts(Bauelement rechts) {
        System.out.println("Ein Abschluss hat kein rechts Element!");
    }
}
```

```
public Bauelement gibLinks() {
    System.out.println("Linkes Element nicht bekannt!");
    return this;
}

public Bauelement gibRechts() {
    System.out.println("Linkes Element nicht bekannt!");
    return this;
}

public Datenelement gibInhalt() {
    return null;
}

public int gibAnzahl() {
    return 0;
}

public void gibAus() {
}
}
```

Code-Beispiel auf Github ansehen: [src/main/java/org/bschlangaul/aufgaben/aud/baum/kompositum/Abschluss.java](https://github.com/bschlangaul/aufgaben/aud/baum/kompositum/Abschluss.java)

```
class Datenknoten extends Bauelement {
    private Bauelement links, rechts;
    private Datenelement inhalt;

    public Datenknoten(Bauelement links, Bauelement rechts, Datenelement
    ↪ inhalt) {
        this.links = links;
        this.rechts = rechts;
        this.inhalt = inhalt;
    }

    public void setztleLinks(Bauelement links) {
        this.links = links;
    }

    public void setzeRechts(Bauelement rechts) {
        this.rechts = rechts;
    }

    public void inhaltSetzen(Datenelement inhalt) {
        this.inhalt = inhalt;
    }

    public Bauelement gibLinks() {
        return links;
    }

    public Bauelement gibRechts() {
        return rechts;
    }
}
```

```
public Datenelement gibInhalt() {
    return inhalt;
}

public int gibAnzahl() {
    return 1 + links.gibAnzahl() + rechts.gibAnzahl();
}

public void gibAus() {
    System.out.print(" [");
    links.gibAus();
    inhalt.gibAus();
    rechts.gibAus();
    System.out.print("] ");
}
}
```

Code-Beispiel auf Github ansehen: [src/main/java/org/bschlangaul/aufgaben/aud/baum/kompositum/Datenknoten.java](#)

```
abstract class Datenelement {
    public abstract void gibAus();
}
```

Code-Beispiel auf Github ansehen: [src/main/java/org/bschlangaul/aufgaben/aud/baum/kompositum/Datenelement.java](#)

```
class Inhalt extends Datenelement {
    private String inhalt;

    public Inhalt(String inhalt) {
        this.inhalt = inhalt;
    }

    public String gibInhalt() {
        return inhalt;
    }

    public void gibAus() {
        System.out.print(inhalt);
    }
}
```

Code-Beispiel auf Github ansehen: [src/main/java/org/bschlangaul/aufgaben/aud/baum/kompositum/Inhalt.java](#)

```
import static org.junit.Assert.assertEquals;

import org.junit.Test;

public class BinaerbaumTest {

    @Test
    public void teste(){
        Binaerbaum baum = new Binaerbaum();
```

```

Inhalt[] inhalte = new Inhalt[16];
Datenknoten[] datenknoten = new Datenknoten[16];
inhalte[0] = new Inhalt("Inhalt 1");

inhalte[1] = new Inhalt("Inhalt 2");
inhalte[2] = new Inhalt("Inhalt 3");

inhalte[3] = new Inhalt("Inhalt 4");
inhalte[4] = new Inhalt("Inhalt 5");

for (int i = 0; i < 5; i++) {
    datenknoten[i] = new Datenknoten(new Abschluss(), new Abschluss(),
    → inhalte[i]);
}
baum.setzeWurzel(datenknoten[0]);

datenknoten[0].setzteLinks(datenknoten[1]);
datenknoten[0].setzeRechts(datenknoten[2]);

datenknoten[1].setzteLinks(datenknoten[3]);
datenknoten[1].setzeRechts(datenknoten[4]);

assertEquals(5, baum.gibAnzahl());

Inhalt inhalt = (Inhalt)
→ baum.gibWurzel().gibLinks().gibLinks().gibInhalt();
assertEquals("Inhalt 4", inhalt.gibInhalt());
}
}

```

Code-Beispiel auf Github ansehen: [src/test/java/org/bschlangaul/aufgaben/aud/baum/kompositum/BinaerbaumTest.java](https://github.com/bschlangaul/aufgaben/aud/baum/kompositum/BinaerbaumTest.java)

## 66115 / 2021 / Frühjahr / Thema 2 / Aufgabe 4

Kreuzen Sie bei der folgenden Multiple-Choice-Frage die richtige(n) Antwort(en) an. Auf falsch gesetzte Kreuze gibt es je einen Minuspunkt. Die Aufgabe wird nicht mit weniger als 0 Punkten gewertet.

(a) Welche Vorgehensmodelle sind für Projekte mit häufigen Änderungen gedacht?

- ☐ EXtreme Programming (XP)
- ☐ Das V-Modell 97
- ☐ Wasserfallmodell
- ☐ Scrum

Lösungsvorschlag

- ☒ EXtreme Programming (XP)
- ☐ Das V-Modell 97
- ☐ Wasserfallmodell

☒ ScrumPrototyping  
Unit-Test  
Anforderungsanalyse

(b) Welche der folgenden Aussagen ist korrekt?

- ☐ Mittels Prototyping versucht man die Anzahl an nötigen Unit-Tests zu reduzieren.
- ☐ Ein Ziel von Prototyping ist die Erhöhung der Qualität während der Anforderungsanalyse.
- ☐ Mit Prototyping versucht man sehr früh Feedback von Stakeholdern zu erhalten.

Lösungsvorschlag

- ☐ Mittels Prototyping versucht man die Anzahl an nötigen Unit-Tests zu reduzieren.
- ☒ Ein Ziel von Prototyping ist die Erhöhung der Qualität während der Anforderungsanalyse.
- ☒ Mit Prototyping versucht man sehr früh Feedback von Stakeholdern zu erhalten.

(c) Welche der folgenden Aussagen ist korrekt?

- ☐ Das Wasserfallmodell sollte nur für große Projekte eingesetzt werden, da der Einarbeitungsaufwand sehr groß ist.
- ☐ Eine gute Anforderungsspezifikation muss vor allem für Ingenieure verständlich sein, da die Anforderungsspezifikation die Grundlage der Systementwicklung bildet.
- ☐ Verifikation ist der Prozess der Beurteilung eines Systems mit dem Ziel festzustellen, ob die spezifizierten Anforderungen erfüllt sind.
- ☐ Durch Validierung kann überprüft werden, ob das Produkt den Erwartungen des Kunden entspricht.
- ☐ Mit Hilfe eines Black-Box-Tests kann man die Korrektheit eines Programmcodes beweisen.

Lösungsvorschlag

- ☐ Das Wasserfallmodell sollte nur für große Projekte eingesetzt werden, da der Einarbeitungsaufwand sehr groß ist.
- ☐ Eine gute Anforderungsspezifikation muss vor allem für Ingenieure verständlich sein, da die Anforderungsspezifikation die Grundlage der Systementwicklung bildet.
- ☐ Verifikation ist der Prozess der Beurteilung eines Systems mit dem Ziel festzustellen, ob die spezifizierten Anforderungen erfüllt sind.
- ☒ Durch Validierung kann überprüft werden, ob das Produkt den Erwartungen des Kunden entspricht.
- ☐ Mit Hilfe eines Black-Box-Tests kann man die Korrektheit eines Programmcodes beweisen.

codes beweisen.

## 66115 / 2021 / Frühjahr / Thema 2 / Aufgabe 4

Das Entwicklerteam von Teacher-Data hat einen neuen Auftrag bekommen. Sie sollen für die Automaten-Videothek DVD Rental die Software entwickeln. Nach den ersten Gesprächen konnten folgende Informationen zusammengetragen werden:

- DVD-Verleih-Automat:
  - Der Kunde soll einfach, schnell und günstig am Automaten 24 Stunden, sieben Tage die Woche DVDs ausleihen können. Der Kunde hat des Weiteren die Möglichkeit, Filme vorab über das Internet zu reservieren. In der DVD-Verleihstation gibt es drei verschiedene Automaten (siehe Zeichnung).
  - Um DVDs ausleihen zu können, müssen sich Neukunden während der Service- und Anmeldezeiten der DVD-Verleihstation bei einem Mitarbeiter anmelden. Dabei werden die benötigten Daten (*Name, Vorname, Geburtsdatum, Anschrift, Personalausweisnummer, Guthaben, Kennwort*) des Kunden aufgenommen und im System gespeichert. Zur persönlichen Identifikation des Kunden wird anschließend der Fingerabdruck am Automaten eingelesen. Der Kunde bekommt eine Magnetkarte ausgehändigt, um sich an den Automaten anmelden zu können.
  - Beim DVD-Ausleih muss sich der Kunde mit der Magnetkarte und seinem Fingerabdruck am Automaten für die Filmauswahl identifizieren. Am Eingabe-Terminal hat der Kunde die Möglichkeit Filme nach *Genre, Schauspieler, Titel, TOP-10, Neuerscheinungen* und *Stichworten* zu suchen. Suchergebnisse werden mit folgenden Informationen auf dem Display präsentiert: *Titel des Films, Regisseur, Hauptdarsteller, Kurzbeschreibung, Erscheinungsjahr, Altersfreigabe und Verfügbarkeit*. Nach der Wahl eines Films, erhält der Kunde seine Magnetkarte zurück und kann den Film am Automaten für die Aus- und Rückgabe nach erneuter Identifikation durch die Karte und den Fingerabdruck abholen. Die DVD wird über den Ausgabeschacht ausgegeben.
  - Bei der Rückgabe einer DVD muss sich der Kunde am Automaten für die Aus- und Rückgabe von Filmen mit der Magnetkarte und seinem Fingerabdruck am Automaten identifizieren. Nach erfolgreicher Identifikation führt der Kunde die DVD in den Rückgabeschacht ein. Der Automat liest daraufhin den Barcode auf der DVD und sortiert diese eigenständig wieder ein und bucht die Leihgebühr vom Guthaben auf der Karte ab.
  - Das Guthaben kann am Automaten zum Aufladen des Guthabens aufgeladen werden. Nach Eingabe der Kundenkarte können Münzen (5, 10, 20, 50 Cent, 1 und 2€) und Scheine (5, 10, 20, 50€) bar eingezahlt werden. Bei einem Betrag von 20€ gibt es einen Bonus von 3€, bei 30€ einen von 5€ und bei einem von 50€ von 10€. Die Karte wird bei beendeter Geldeingabe nach Drücken des roten Knopfes wieder ausgeworfen.

- Reservierung von Filmen über das Internet: Nach erfolgreicher Anmeldung mit Kundennummer und Geheimzahl auf der betreibereigenen Homepage kann der Kunde nach *Genre, Schauspieler, Titel, TOP-10, Neuerscheinungen* und *Stichworten* Filme suchen. Suchergebnisse werden mit folgenden Informationen präsentiert: *Titel des Films, Regisseur, Hauptdarsteller, Kurzbeschreibung, Erscheinungsjahr, Altersfreigabe* und *Verfügbarkeit*. Der Kunde kann nach der Suche einzelne Filme reservieren, die er innerhalb von zwei Stunden abholen muss. Sollte dies nicht in dem Zeitraum geschehen, so werden die Filme nach zwei Stunden wieder zum Ausleihen freigegeben.
- Weitere Rahmenbedingungen:
  - Wird die Kundenkarte im Automaten vergessen, so wird sie zur Sicherheit automatisch eingezogen und kann während der Servicezeiten wieder abgeholt werden.
  - Die maximale Leihdauer beträgt 10 Tage.
  - Hat der Kunde versehentlich den falschen Film ausgeliehen, so kann er diese bis 10 Minuten nach Ausgabe der DVD ohne Berechnung einer Leihgebühr wieder am Automaten zurückgeben.
  - Die Leihgebühr für jeden angefangenen Tag (24h) beträgt 2€.
  - Zu jedem Film gibt es mehrere Exemplare.
  - Die Ausgabe und Einsortierung von DVDs übernimmt ein DVD-Archiv-Roboter, der über eine Schnittstelle angesteuert werden muss.

(a) Identifizieren Sie die Akteure des oben beschriebenen Systems.

Lösungsvorschlag

Kunde, Service-Mitarbeiter, Wartung, DVD-Archiv-Roboter

(b) Identifizieren Sie die nicht-funktionalen und funktionalen Anforderungen (als Use Cases).

Lösungsvorschlag

**Nicht-funktional:** Bedienoberfläche Web / Automat gleich, einfache Handhabbarkeit, schnell, Dauerbetrieb, Ausfallquote, Leihdauer

**Funktional:** Geldkarte aufladen, Film suchen, Film auswählen, Film entnehmen, Film zurückgeben, Kunden aufnehmen, Film reservieren

(c) Geben Sie eine formale Beschreibung für drei Use Cases an. Orientieren können Sie sich an folgendem Beispiel:

Lösungsvorschlag

**Geldkarte aufladen**



Use Case:	Geldkarte aufladen
Ziel:	Auf der Geldkarte befindet sich ein gewünschter Betrag an Geld.
Kategorie:	primär
Vorbedingung:	Geldkarte befindet sich im Automaten.
Nachbedingung Erfolg:	Auf der Geldkarte befindet sich der gewünschte Betrag.
Nachbedingung Fehlschlag:	Betrag auf der Karte ist der gleiche wie davor → Fehlermeldung
Akteure:	Kunde
Auslösendes Ereignis:	Geldkarte wird in den Automaten zum Aufladen des Guthabens geschoben.
Beschreibung:	<ul style="list-style-type: none"> <li>(i) Kunde schiebt Karte in den Automaten.</li> <li>(ii) Kunde wirft Münzen oder gibt einen Geldschein ein.</li> <li>(iii) Kunde drückt Knopf zum Auswerfen der Karte.</li> <li>(iv) Karte wird ausgegeben.</li> </ul>
Erweiterungen:	Schein oder Münze wird nicht akzeptiert. → Fehlermeldung
Alternativen:	Mindest-Guthaben auf der Karte

### Film suchen

Use Case:	Film suchen
Ziel:	Anzeigen des gesuchten Films
Kategorie:	primär
Vorbedingung:	Kunde hat sich mit Magnetkarte und Fingerabdruck identifiziert.
Nachbedingung Erfolg:	Eine Liste von Filmen wird angezeigt.
Nachbedingung Fehlschlag:	Kein Film gefunden → Fehlermeldung
Akteure:	Kunde
Auslösendes Ereignis:	Kunde will einen Film ausleihen.
Beschreibung:	Kunde gibt Genre, Schauspieler, Titel oder Stichwort ein oder lässt sich Neuerscheinungen und TOP10 auflisten.
Erweiterungen:	–
Alternativen:	Vorgaben von weiteren Suchkriterien

**Kunden aufnehmen**

Use Case:	Kunden aufnehmen
Ziel:	Kunde ist in der Kundendatei.
Kategorie:	primär
Vorbedingung:	Kunde ist noch nicht in der Kundendatei vorhanden.
Nachbedingung Erfolg:	Kunde ist als Mitglied in der Kundendatei aufgenommen.
Nachbedingung Fehlschlag:	Kunde kann nicht in die Datei aufgenommen werden.
Akteure:	Kunde, Service-Mitarbeiter
Auslösendes Ereignis:	Kunde stellt zu den Öffnungszeiten bei einem Service-Mitarbeiter einen Mitgliedsantrag.
Beschreibung:	<ul style="list-style-type: none"> <li>(i) Kunde meldet sich zu den Öffnungszeiten bei einem Service-Mitarbeiter an.</li> <li>(ii) Service-Mitarbeiter erfasst nötige Daten (Name, Vorname, Geburtsdatum, Anschrift, Personalausweisnummer, Kennwort).</li> <li>(iii) Service-Mitarbeiter kassiert einen Startbetrag und erfasst ihn bei den Kundendaten.</li> <li>(iv) Der Fingerabdruck des Kunden wird erfasst und gespeichert.</li> <li>(v) Kunde bekommt eine Magnetkarte mit seinen Daten.</li> </ul>
Erweiterungen:	-
Alternativen:	Kunde ist bereits im System und lässt nur die Kundendaten ändern.

- (d) Verfeinern Sie nun die Use Cases, indem Sie ähnliche Teile identifizieren und daraus weitere Use Cases bilden.

Lösungsvorschlag

Identifizierung des Kunden an den Automaten, Filmsuche

- (e) Nennen Sie mindestens drei offene Fragen, die in einem weiteren Gespräch mit dem Kunden noch geklärt werden müssten.<sup>3</sup>

<sup>3</sup>Quelle: Universität Stuttgart, Institut für Automatisierungs- und Softwaretechnik Prof. Dr.-Ing. Dr. h. c. P. Göhner

- Was passiert nach der maximlaen Leihdauer?
- Was passiert, wenn der Finger nicht zur Karte passt?
- Was passiert mit defekten DVDs?
- Was passiert bei Rückgabe der falschen DVD?

## Wasserfallmodell

Die Firma Teacher-Data soll für einen Auftraggeber ein Informationssystem entwickeln. Das Entwicklerteam entscheidet sich, nach dem *Wasserfallmodell* vorzugehen.

- (a) Geben Sie an, welche *Phasen* dabei durchlaufen werden und erläutern Sie kurz deren *Zweck / Intention / Aufgabe*.

(i) Problem- und Systemanalyse

Grobe Formulierung der Ziele (z. B. Einsatzplattform Windows, einheitliche Software zur Unterstützung des Verkaufs, ...)

- Erfassung des Problembereichs (z. B. Interaktion zwischen Händler und Hersteller soll komplett über das Netz funktionieren.)
- Machbarkeitsstudie (z. B. Händler sind bereit, sich an das zu entwickelnde System anzupassen.)
- Analyse des Ist-Zustandes (z. B. Händler A bietet Interface X wohingegen Händler B Interface Z bietet. Es wird bereits teilweise bei Händler C über das Netz bestellt. ...)
- Anforderungsspezifikation (Lastenheft) (z. B. Einheitliche Oberfläche für alle Benutzer beim Hersteller, möglichst wenige Veränderungen für die Mitarbeiter (= schnelle Eingewöhnungsphase) ...)
- Systemspezifikation (Pflichtenheft) (z. B. soll der Ablauf eines Kaufvorgangs folgendermaßen aussehen: Händlerauswahl, Bestellung, Abrechnung ... Die Benutzeroberfläche soll folgende Informationen anzeigen: Lieferstatus, ....)

(ii) Systementwurf

- Systemarchitektur (z. B. Klassendiagramm, Festlegung von Komponenten ...)
- Schnittstellen der Komponenten festlegen (z. B. Datenbank, Middleware, ...)
- Modulkonzept (z. B. Modulstruktur und Art der Module (funktional, prozedural, objektorientiert))
- Definition der Einzelschnittstellen (z. B. Interfaces und abstrakte Klassen in Java)

(iii) Implementierung

- Strukturierung der Komponenten in Module (z. B. Klassenrumpfe)
- Spezifikation einzelner Module (z. B. exakte Beschreibung eines Moduls)
- Codierung, Generierung, Wiederverwendung einzelner Module (z. B. Methoden in Klassen, ...)

(iv) Integration und Test

- Integration von Modulen (z. B. Zusammensetzen von Datenbank und Applikation zum Server)
- Integration von Komponenten (z. B. Einsetzen von Client und Server ins Gesamtsystem.)
- Testen und Verifikation (z. B. Testen, ob die Implementierung zusammen mit der Umgebung funktioniert. Zusichern, dass gewisse Eigenschaften gültig sind.)

(v) Wartung und Weiterentwicklung

- Wartung (z. B. Korrektur eines Fehlers im Programm, Portierung auf anderes Betriebssystem, ...)
- Erweitern der Funktionalität (z. B. Einbeziehen einer neuen Option im Geschäftsprozess)
- Anpassung der Funktionalität (z. B. Anpassung an Änderung im Geschäftsprozess)

(b) Während des Entwicklungsprozesses werden nach und nach im Folgenden aufgelistete Teilprodukte entstehen. Ordnen Sie diese den jeweiligen Phasen zu.

- Schätzung der Entwicklungskosten
- Dokumentation der Änderungen nach Produktabnahme
- Pflichtenheft
- Beschreibung der Datenstruktur in der Datenbank
- Integration von Modulen/Komponenten
- Betriebsbereites Informationssystem
- Beschreibung der Schnittstelle einzelner Softwarekomponenten
- Quellcode für die GUI
- Durchführbarkeitsstudie
- Systemarchitektur

Lösungsvorschlag

- Schätzung der Entwicklungskosten → *Anforderungsdefinition*
- Dokumentation der Änderungen nach Produktabnahme → *Betrieb und Wartung*
- Pflichtenheft → *Anforderungsdefinition*

- Beschreibung der Datenstruktur in der Datenbank → *System- und Softwareentwurf*
- Integration von Modulen/Komponenten → *Integration und Systemtest*
- Betriebsbereites Informationssystem → *Betrieb und Wartung*
- Beschreibung der Schnittstelle einzelner Softwarekomponenten → *System- und Softwareentwurf*
- Quellcode für die GUI → *Implementierung und Komponententest*
- Durchführbarkeitsstudie → *Anforderungsdefinition*
- Systemarchitektur → *System- und Softwareentwurf*

Wasserfallmodell  
Spiralmodell  
V-Modell  
Evolutionäre  
Softwaremodelle  
Inkrementelle  
Prozessmodelle  
SCRUM

## Vorgehensmodelle

Dem Team von Teacher-Data kommen Zweifel, ob die Entscheidung für das Wasserfallmodell für die Umsetzung des Informationssystems richtig war, oder ob ein anderes Vorgehen eventuell besser wäre. Beurteilen und vergleichen Sie die Vorgehensmodelle

- Wasserfallmodell
- Spiralmodell
- V-Modell
- Evolutionäres Modell oder Inkrementelles Modell
- agile Entwicklung, wie z. B. Scrum

nach den folgenden Gesichtspunkten:

- (a) Größe des Entwicklerteams
- (b) Komplexität des Projekts
- (c) Bekanntheit der Anforderungen
- (d) Änderung der Anforderungen
- (e) Zeitspielraum (Time-to-Market; was muss bis wann fertig sein?)
- (f) Dokumentation
- (g) IT-Kenntnisse des Kunden
- (h) Durchschnittliche Anzahl an Iterationen

Lösungsvorschlag

	Wasserfallmodell	Spiralmodell	V-Modell	evolutionär / inkrementell	agil
Größe des Entwicklerteams	diskutiert die Projektgröße nicht	mittlere Teams	mittlere und große Teams	diskutiert die Projektgröße nicht	ca. 3-9 Entwickler (ohne Product Owner und Scrum Master)
Komplexität des Projekts	einfache Projekte mit stabilen Anforderungen	einfache Projekte mit stabilen Anforderungen; Anforderungen können jedoch in den Iterationen angepasst werden.	komplexe Projekte; bei einfachen Projekten relativ viel bürokratischer Overhead.	große, lange und komplexe Projekte	hohe Komplexität möglich
Bekanntheit der Anforderungen	Alle Anforderungen müssen zu Beginn des Projekts bekannt sein.	Es muss eine initiale Menge von Anforderung bekannt sein. Der Kunde kann die Anforderungen aber aufgrund von Prototypen erweitern.	evolutionär: Alle Anforderungen müssen zu Beginn des Projekts bekannt sein. inkrementell: Anforderungen müssen nicht von Anfang an bekannt sein.	Anforderungen müssen von Anfang an bekannt sein.	Es muss eine initiale Menge von Anforderung bekannt sein. Der Kunde kann die Anforderungen aber laufend erweitern / ggf. ändern.
Änderungen der Anforderungen	Schwer möglich. Durch den strikten Top-Down-Ansatz von der Anforderung hin zum Code zieht eine Änderung einen tiefen Eingriff in den Prozess mit sich.	Gut möglich. In jeder Runde der Spirale können die Anforderungen neu definiert werden.	Schwer möglich → strikter Top-Down-Ansatz	evolutionär: häufige Änderungen möglich. inkrementell: Änderungen teilweise möglich.	rasche Anpassung an neue Anforderungen möglich (deshalb agil!)
Zeitspielraum	Wenig Spielraum. Üblicherweise muss der ganze Prozess durchlaufen werden. Es ist höchstens möglich, den Prozess zu beschleunigen, indem man Anforderungen streicht. Es wird also alles zum Schluss fertig.	Relativ flexibel. Sobald ein Prototyp existiert, der akzeptabel ist, kann dieser auf den Markt gebracht werden.	Wenig Spielraum (vgl. Wasserfallmodell)	Einsatzfähige Produkte in kurzen Zeitabständen, daher relativ flexibel.	Unterteilung in Sprints, daher schnell lauffähige Prototypen vorhanden. Bei der Planung des nächsten Sprints kann auf neue zeitliche Gegebenheiten relativ flexibel eingegangen werden.
Dokumentation	Viel Dokumentation – Für jede Phase wird eine komplette Dokumentation erstellt.	Sehr viel Dokumentation – In jedem Zyklus werden alle Phasen (inkl. Dokumentation) durchlaufen, aber nur im Umfang des Prototypen. Es werden alle Artefakte und Tätigkeiten dokumentiert.	Viel Dokumentation – Gerade in den ersten Phasen wird sehr viel über die Software schriftlich festgehalten. Aber auch für die anderen Phasen wird eine komplette Dokumentation erstellt.		Anforderungsdokumentation ist sehr wichtig. Ansonsten ist funktionierende Software höher zu bewerten als eine umfangreiche Dokumentation.
IT-Kenntnisse des Kunden	Wenig Kenntnisse nötig. Meistens schreibt der Kunde das Lastenheft und der Entwickler versucht dieses dann in einem Pflichtenheft umzusetzen. Wenn das Lastenheft vom Entwickler als machbar befunden wird, bekommt der Kunde das fertige Produkt.	Wenig Kenntnisse nötig. – Kunde sieht immer nur die fertigen Prototypen und äußert dann seine Wünsche.	Wenig Kenntnisse nötig (vgl. Wasserfallmodell)	Wenig Kenntnisse nötig	Wenig Kenntnisse nötig, aber von Vorteil, da enge Zusammenarbeit mit dem Kunden
Durchschnittliche Anzahl an Iterationen	vgl. V-Modell	ca. 3-5 Iterationen	Nur lange Zyklen. – Mit dem V-Modell sind nur sehr lange Zyklen handhabbar, da immer wieder der ganze Entwicklungsprozess durchlaufen werden muss.		variable Anzahl an Sprints, je nach Projektgröße

Zustandsdiagramm zeichnen  
Aktivitätsdiagramm

## 66115 / 2021 / Frühjahr / Thema 2 / Aufgabe 4

- (a) Beschreiben Sie den Unterschied zwischen einem Zustands- und einem Aktivitätsdiagramm.

Das Aktivitätsdiagramm zeigt den Fluss von Aktivitäten innerhalb des Systems. Das Zustandsdiagramm hingegen beschreibt die Reaktion eines Systems auf Ereignisse.

- (b) Der DVD-Automat für die Filmauswahl aus Blatt 2, Aufgabe 1 soll als Zustandsdiagramm modelliert werden. Beachten Sie dabei die angegebenen Funktionalitäten des Automaten.
- (i) Geben Sie die Ein- und Ausgaben des Automaten für die Filmauswahl und für die Aus- und Rückgabe von Filmen an.

Lösungsvorschlag

**Eingaben:** Kundenkarte, Fingerabdruck, DVDs, Suchbegriff

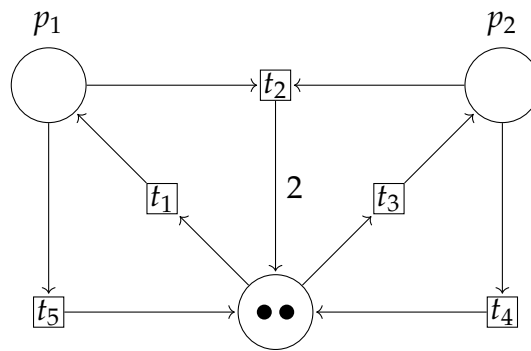
**Ausgaben:** Kundenkarte, DVDs, Bildschirmausgaben (Filmtitel, Regisseur, Hauptdarsteller, Kurzbeschreibung, Erscheinungsjahr, Altersfreigabe, Verfügbarkeit, Fehlermeldung etc.)

- (ii) Geben Sie alle Zustandsattribute an, die für die Modellierung der Automaten notwendig sind und beschreiben Sie deren Verwendungszweck.
- (iii) Identifizieren Sie anhand der Zustandsattribute die Zustände der obigen Automaten und geben Sie eine Charakterisierung der Zustände durch Angabe der möglichen Wertebereiche der Zustandsattribute an. Welcher der Zustände ist der Anfangszustand?
- (iv) Zeichnen Sie die Zustandsübergangsdiagramme. Verwenden Sie hierzu die Syntax mit Ein- und Ausgabe, Vor- und Nachbedingungen.
- (c) Betrachten wir nun einen gewöhnlichen Video- und DVD-Verleih. Beschreiben Sie das Szenario des Ausleihs eines Videos mit Hilfe eines Aktivitätsdiagramms. Bei dem Videoverleih gelte:
- Der Kunde identifiziert sich beim Ausleihen mit seiner Kundenkarte oder seinem Passwort. Hat der Kunde noch keine Karte, so muss der Mitarbeiter ihn registrieren und ihm eine Kundenkarte ausstellen.
  - Filme können wie beim Automaten per Internet bis zu zwei Stunden im Voraus reserviert werden.
  - Der Kunde hat kein Gehaltskonto, sondern bezahlt seine Gebühren bei der Rückgabe des Videos in bar oder per Karte.
  - Ansonsten gelten die gleichen Bedingungen wie beim Automaten.

## 66115 / 2021 / Frühjahr / Thema 2 / Aufgabe 4

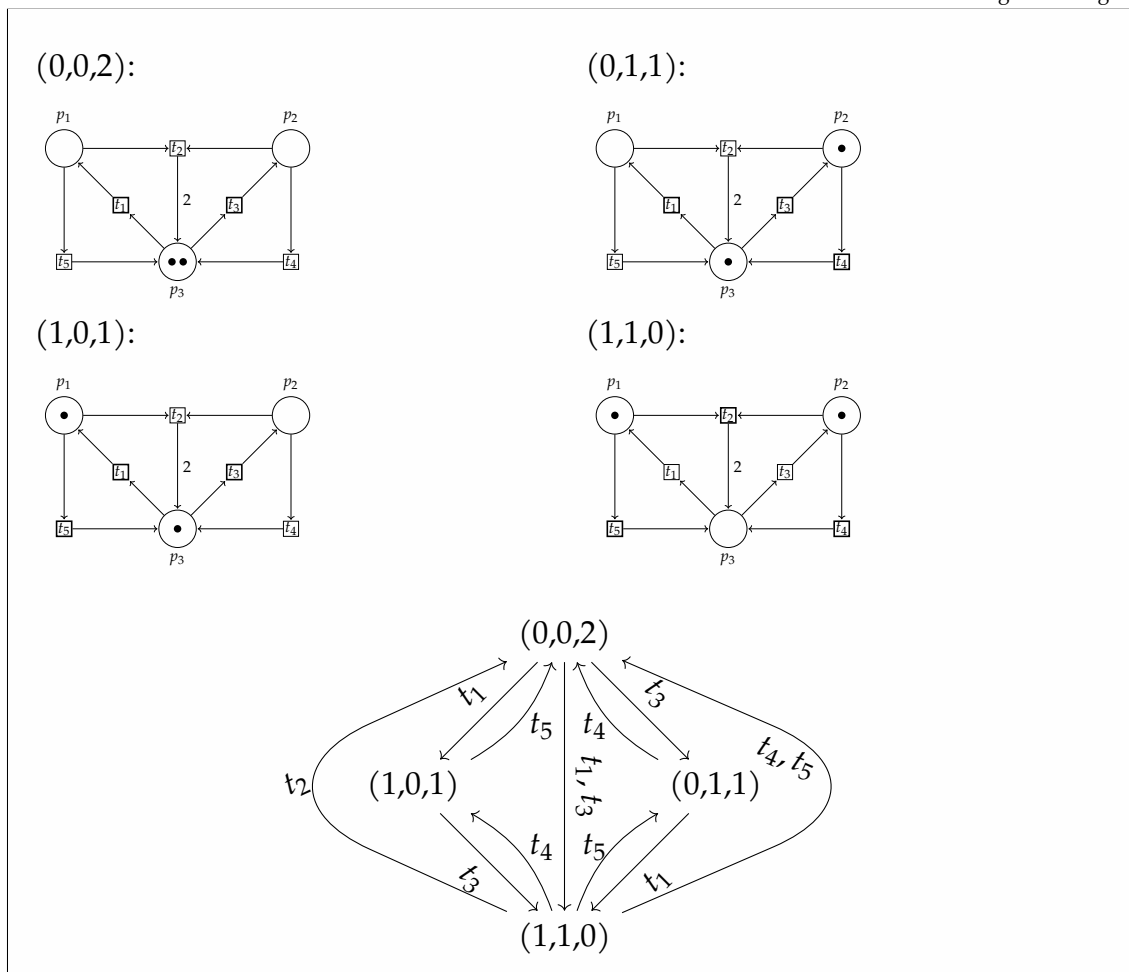
Gegeben Sei das folgende Petri-Netz:





(a) Zeichnen Sie den Erreichbarkeitsgraphen des Petri-Netzes.

Lösungsvorschlag



(b) Begründen Sie anhand des Erreichbarkeitsgraphen, ob das Petri-Netz lebendig, umkehrbar und/oder verklemmungsfrei ist.

Lösungsvorschlag

**lebendig** Ja. Es gibt im Erreichbarkeitsgraphen keine Senke, also keinen Zustand, aus dem man nicht mehr heraus kommt.

**umkehrbar** Im Erreichbarkeitsgraphen kommt man von  $(0,0,2)$  auf verschiedenen Wegen wieder zurück zu  $(0,0,2)$ . Man kann sich unendlich oft im

Graph bewegen. Die Anfangsmarkierung kann wieder erreicht werden ( $t_1 \rightarrow t_3 \rightarrow t_2$  oder  $t_1 \rightarrow t_3 \rightarrow t_5 \rightarrow t_4$ ).

**verklemmungsfrei** Ja. Es gibt im Erreichbarkeitsgraphen keine Senke, also keinen Zustand, aus dem man nicht mehr herauskommt.

- (c) Geben Sie die Darstellungsmatrix  $A$  sowie den Belegungsvektor  $v$  an und berechnen Sie damit die Belegung nach Schaltung von  $t_1 \rightarrow t_3 \rightarrow t_2$ .

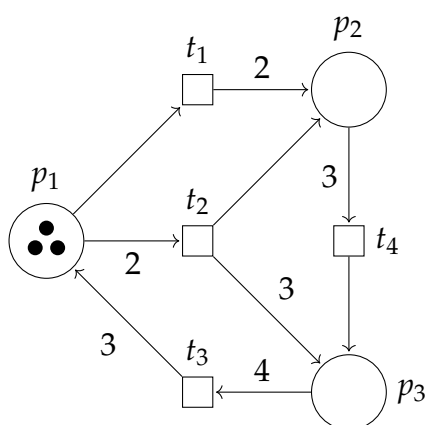
Lösungsvorschlag

$$A = \begin{matrix} & t_1 & t_2 & t_3 & t_4 & t_5 \\ \begin{matrix} p_1 \\ p_2 \\ p_3 \end{matrix} & \begin{pmatrix} 1 & -1 & 0 & 0 & -1 \\ 0 & -1 & 1 & -1 & 0 \\ -1 & 2 & -1 & 1 & 1 \end{pmatrix} \end{matrix}, \quad v = \begin{pmatrix} 0 \\ 0 \\ 2 \end{pmatrix}$$

$$v_{\text{neu}} = v + A \cdot \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix} + A \cdot \begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \\ 0 \end{pmatrix} + A \cdot \begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \\ 0 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 2 \end{pmatrix} = v$$

## 66115 / 2021 / Frühjahr / Thema 2 / Aufgabe 4

Gegeben ist das folgende Petri-Netz:



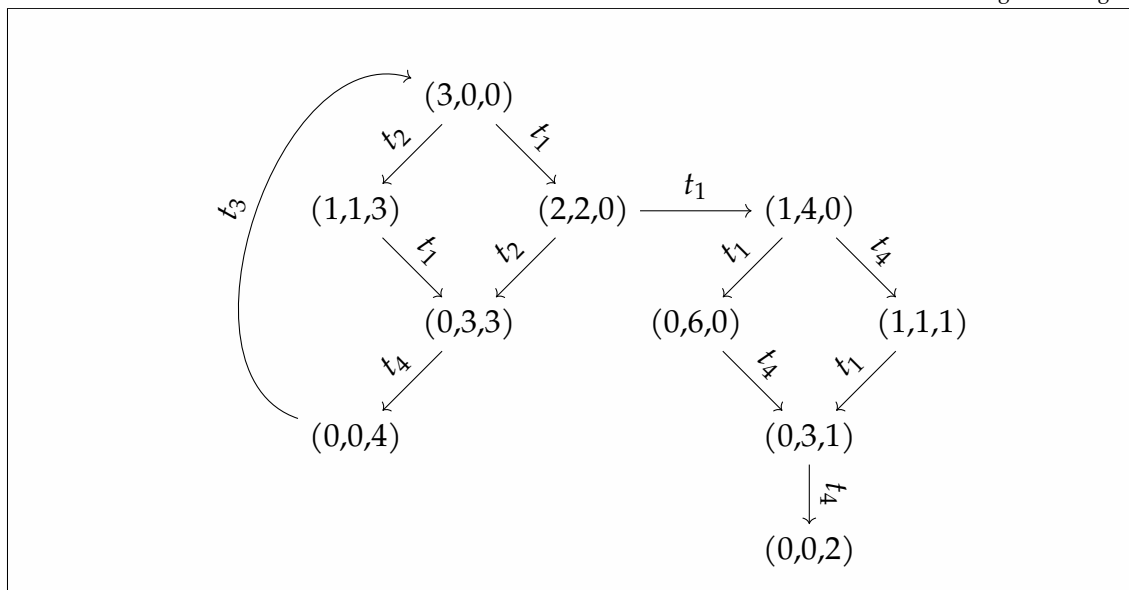
- (a) Geben Sie die dazugehörige Darstellungsmatrix sowie den Belegungsvektor an.

Lösungsvorschlag

$$A = \begin{matrix} & t_1 & t_2 & t_3 & t_4 \\ \begin{matrix} p_1 \\ p_2 \\ p_3 \end{matrix} & \begin{pmatrix} -1 & -2 & 3 & 0 \\ 2 & 1 & 0 & -3 \\ 0 & 3 & -4 & 1 \end{pmatrix} \end{matrix}, \quad v = \begin{pmatrix} 3 \\ 0 \\ 0 \end{pmatrix}$$

(b) Skizzieren Sie den Erreichbarkeitsgraphen des Petri-Netzes.

Lösungsvorschlag



(c) Begründen Sie anhand des Erreichbarkeitsgraphen, ob das Petri-Netz verklemmungsfrei ist oder nicht.

Lösungsvorschlag

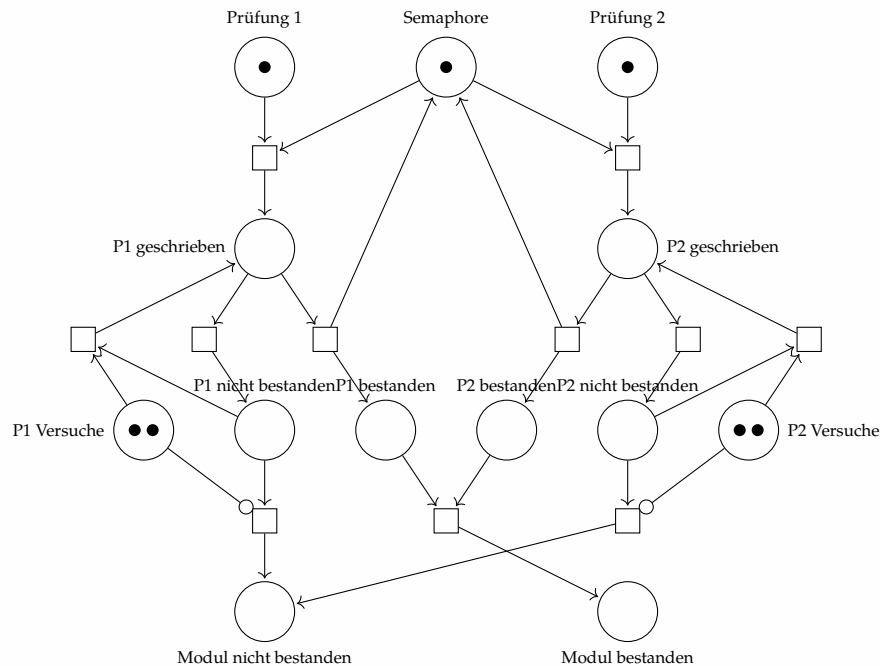
Durch Schalten von  $t_1 \rightarrow t_1 \rightarrow t_1 \rightarrow t_4 \rightarrow t_4$  wird beispielsweise eine Verklemmung erreicht. Das Petri-Netz ist also nicht verklemmungsfrei. Am Erreichbarkeitsgraphen erkennt man das anhand der Senke im Knoten  $[0,0,2]$ .

## 66115 / 2021 / Frühjahr / Thema 2 / Aufgabe 4

Modellieren Sie folgendes Szenario als Petri-Netz:

Ein Modul gilt als bestanden, wenn beide Prüfungen  $P_1$  und  $P_2$  bestanden sind. Beide Prüfungen dürfen bei Nicht-Bestehen jeweils maximal zwei mal wiederholt werden. Die Prüfungen dürfen nicht gleichzeitig geschrieben werden. Erst wenn eine von beiden bestanden wurde, darf die nächste begonnen werden. Wurde eine der beiden Prüfungen insgesamt drei mal nicht bestanden, so gilt das gesamte Modul als nicht bestanden.

Man beachte die sog. Inhibitorischen Linien mit Punkt am Ende). Z. B. kann  $t_7$  nur schalten, wenn die Stelle „Wdh. Versuche P1“ keine Markierung enthält.



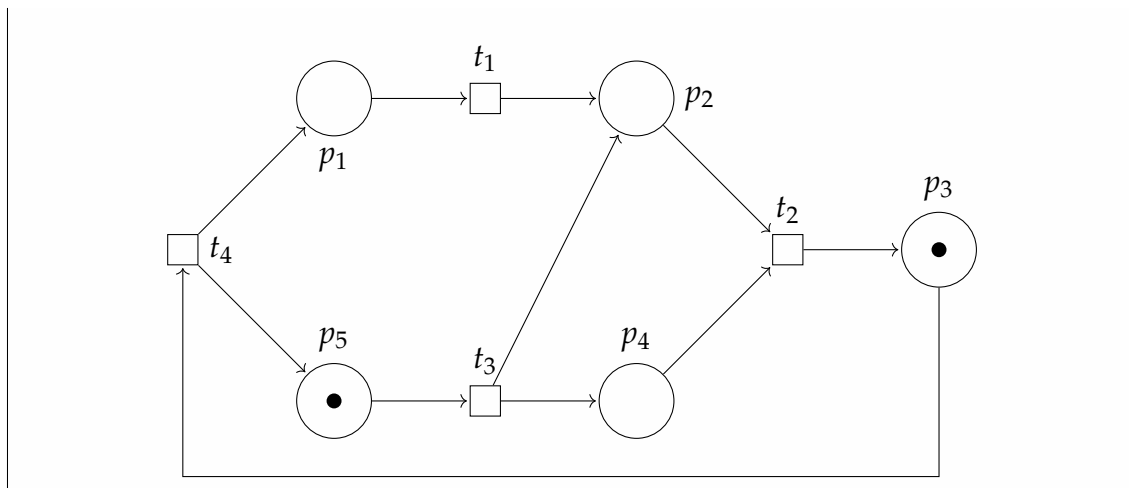
## 66115 / 2021 / Frühjahr / Thema 2 / Aufgabe 4

Gegeben sei die Darstellungsmatrix  $A$  und der Belegungsvektor  $v$  eines Petri-Netzes:

$$A = \begin{matrix} & t_1 & t_2 & t_3 & t_4 \\ \begin{matrix} p_1 \\ p_2 \\ p_3 \\ p_4 \\ p_5 \end{matrix} & \begin{pmatrix} -1 & 0 & 0 & 1 \\ 1 & -1 & 1 & 0 \\ 0 & 1 & 0 & -1 \\ 0 & -1 & 1 & 0 \\ 0 & 0 & -1 & 1 \end{pmatrix} \end{matrix}, \quad v = \begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \\ 1 \end{pmatrix}$$

(a) Skizzieren Sie das zugehörige Petri-Netz.

Lösungsvorschlag



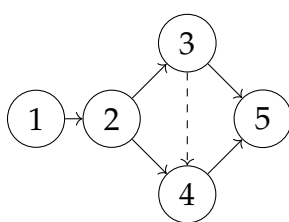
- (b) Berechnen Sie mithilfe der Darstellungsmatrix  $A$  und zum Belegungsvektor  $v$ , die Belegung nach Schaltung von  $t_3 \rightarrow t_2 \rightarrow t_4$ .

Lösungsvorschlag

$$v_{\text{neu}} = v + A \cdot \begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \end{pmatrix} + A \cdot \begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \end{pmatrix} + A \cdot \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \\ 1 \\ 0 \\ 1 \end{pmatrix}$$

## 66115 / 2021 / Frühjahr / Thema 2 / Aufgabe 4

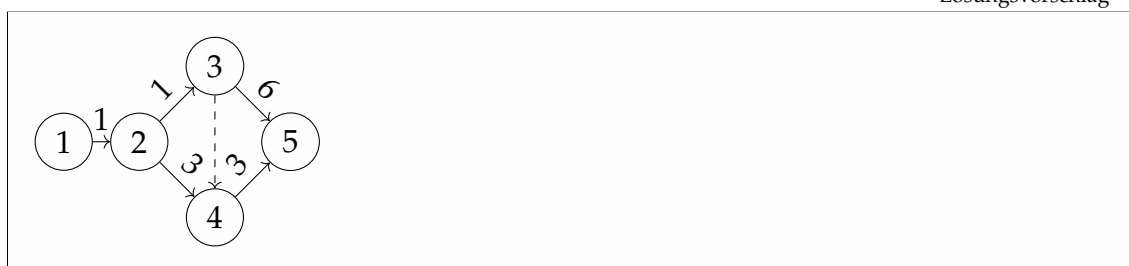
- (a) Gegeben ist folgender (unvollständiger) CPM-Netzplan, sowie die frühesten und spätesten Termine und die Pufferzeiten aller Ereignisse:



Ereignis	1	2	3	4	5
frühester Termin	0	1	2	4	8
spätester Termin	0	1	2	5	8
Puffer	0	0	0	1	0

Vervollständigen Sie den CPM-Netzplan, indem Sie mit Hilfe obiger Tabelle die Zeiten der Vorgänge berechnen.

Lösungsvorschlag



### Frühester Termin/Zeitpunkt

— Wir führen eine Vorwärtsterminierung durch und addieren die Dauern. Kann ein Ereignis über mehrere Vorgänge erreicht werden, wählen wir das Maximum aus. **Erläuterungen:**  $i$ : Ereignis  $i$ ;  $FZ_i$ : Frühester Zeitpunkt, zu dem Ereignis  $i$  eintreten kann.

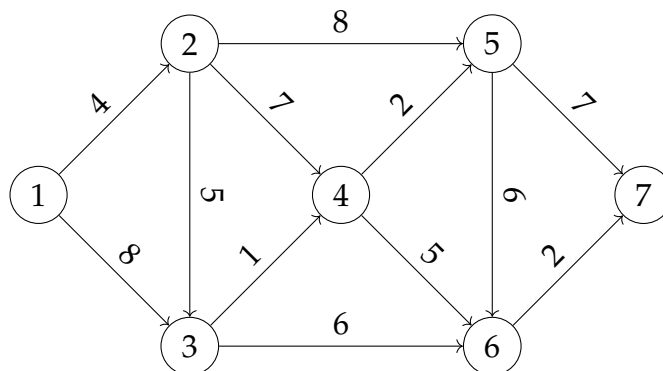
$i$	Nebenrechnung	$FZ_i$
1		0
2		1
3		2
4	$\max(4_2, 2_3)$	4
5	$\max(8_3, 7_4)$	8

### Spätester Termin/Zeitpunkt

— Wir führen eine Rückwärtsterminierung durch und subtrahieren die Dauern vom letzten Ereignis aus. Kann ein Ereignis über mehrere Vorgänge erreicht werden, wählen wir das Minimum aus. **Erläuterungen:**  $i$ : Ereignis  $i$ ;  $SZ_i$ : Spätester Zeitpunkt, zu dem Ereignis  $i$  eintreten kann.

$i$	Nebenrechnung	$SZ_i$
5	siehe $FZ_5$	8
4		5
3	$\min(2_5, 5_4)$	2
2	$\min(1_3, 2_4)$	1
1		0

- (b) Bestimmen Sie zum nachfolgenden CPM-Netzplan für jedes Ereignis den *frühesten Termin*, den *spätesten Termin* sowie die *Gesamtpufferzeit*. Geben Sie außerdem den *kritischen Pfad* an.



$i$	1	2	3	4	5	6	7
$FZ_i$	0	4	9	11	13	19	21
$SZ_i$	0	4	10	11	13	19	21
GP	0	0	1	0	0	0	0

**Frühester Termin/Zeitpunkt**

$i$	Nebenrechnung	$FZ_i$
1		0
2		4
3	$\max(8, 4_{(\rightarrow 2)} + 5) = \max(8, 9)$	9
4	$\max(9_{(\rightarrow 3)} + 1, 4_{(\rightarrow 2)} + 7) = \max(10, 11)$	11
5	$\max(4_{(\rightarrow 2)} + 8, 11_{(\rightarrow 4)} + 2) = \max(12, 13)$	13
6	$\max(13_{(\rightarrow 5)} + 6, 11_{(\rightarrow 4)} + 5, 9_{(\rightarrow 3)} + 6) = \max(19, 16, 15)$	19
7	$\max(13_{(\rightarrow 5)} + 7, 19_{(\rightarrow 6)} + 2) = \max(20, 21)$	21

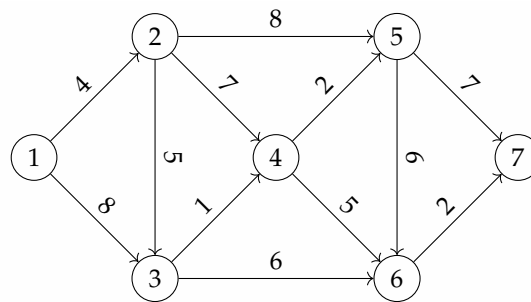
**Spätester Termin/Zeitpunkt**

$i$	Nebenrechnung	$SZ_i$
1	$\min(4_{(\rightarrow 2)} - 4, 10_{(\rightarrow 3)} - 8) = \min(0, 2)$	0
2	$\min(13_{(\rightarrow 5)} - 8, 11_{(\rightarrow 4)} - 7, 10_{(\rightarrow 3)} - 5) = \min(5, 4, 5)$	4
3	$\min(11_{(\rightarrow 4)} - 1, 19_{(\rightarrow 6)} - 6) = \min(10, 13)$	10
4	$\min(13_{(\rightarrow 5)} - 2, 19_{(\rightarrow 6)} - 5) = \min(11, 14)$	11
5	$\min(21_{(\rightarrow 7)} - 7, 19_{(\rightarrow 6)} - 6) = \min(14, 13)$	13
6	$21_{(\rightarrow 7)} - 2$	19
7	siehe $FZ_7$	21

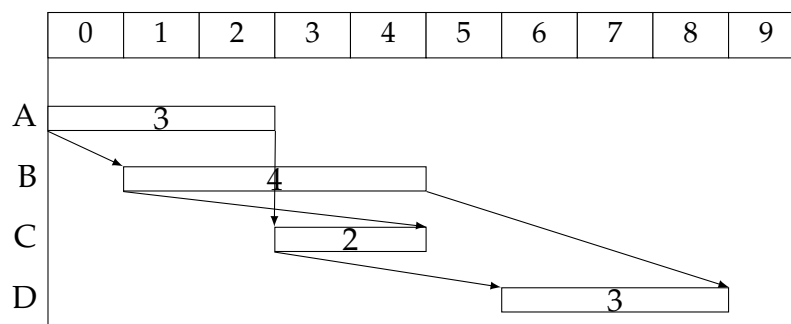
**Kritischer Pfad**

$1 \rightarrow 2 \rightarrow 4 \rightarrow 5 \rightarrow 6 \rightarrow 7$

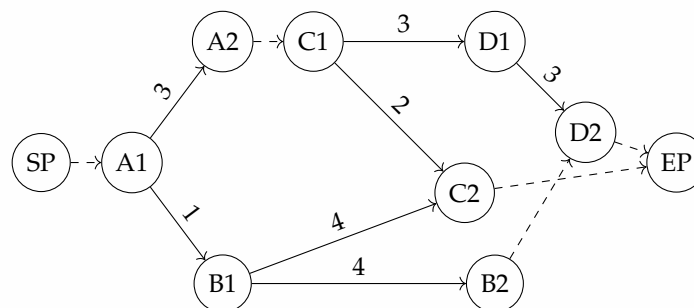
$$4_{(1 \rightarrow 2)} + 7_{(2 \rightarrow 4)} + 2_{(4 \rightarrow 5)} + 6_{(5 \rightarrow 6)} + 2_{(6 \rightarrow 7)} = 21$$



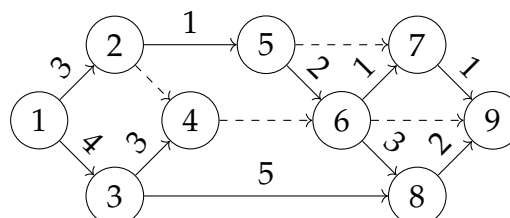
- (c) Konvertieren Sie das nachfolgende Gantt-Diagramm in ein CPM-Netzwerk. Als Hilfestellung ist die Anordnung der Ereignisse bereits vorgegeben.



Lösungsvorschlag



## 66115 / 2021 / Frühjahr / Thema 2 / Aufgabe 4



- (a) Welche Scheinvorgänge könnten aus dem Netzwerk entfernt werden, ohne dass Informationen verloren gehen?

☒  $2 \rightarrow 4$



☐  $4 \rightarrow 6$

☒  $5 \rightarrow 7$

☒  $6 \rightarrow 9$

- (b) Berechnen Sie für jedes Ereignis den frühesten Termin, wobei angenommen wird, dass das Projekt zum Zeitpunkt 0 startet.

Lösungsvorschlag

$i$	Nebenrechnung	$FZ_i$
1		0
2	$0 + 3_{(1 \rightarrow 2)} = 3$	3
3	$0 + 4_{(1 \rightarrow 3)} = 4$	4
4	$3_{(1 \rightarrow 2)} + 0_{(2 \rightarrow 4)} = 3$ $4_{(1 \rightarrow 3)} + 3_{(3 \rightarrow 4)} = 7$ $\max(3, 7)$	7
5	$3_{(1 \rightarrow 2)} + 1_{(2 \rightarrow 5)} = 4$	4
6	$\max(7 + 0, 4 + 2)$	7
7	$\max(4 + 0, 7 + 1)$	8
8	$\max(4 + 5, 7 + 3)$	10
9	$\max(8 + 1, 7 + 0, 10 + 2)$	12

- (c) Berechnen Sie für jedes Ereignis auch die spätesten Zeiten, indem Sie für das letzte Ereignis den frühesten Termin als spätesten Termin ansetzen.

Lösungsvorschlag

$i$	Nebenrechnung	$SZ_i$
1	$\min(4 - 3, 4 - 4)$	0
2	$\min(5 - 1, 7 - 0)$	4
3	$\min(10 - 5, 7 - 3)$	4
4	$7 - 0$	7
5	$\min(11 - 0, 7 - 2)$	5
6	$\min(12 - 0, 11 - 1, 10 - 3)$	7
7	$12 - 1$	11
8	$12 - 2$	10
9	siehe $FZ_9$	12

- (d) Geben Sie nun die Pufferzeiten der Ereignisse an.

Ereignis	1	2	3	4	5	6	7	8	9
frühester Termin	0	3	4	7	4	7	8	10	12
spätester Termin	0	4	4	7	5	7	11	10	12
Puffer	0	1	0	0	1	0	3	0	0

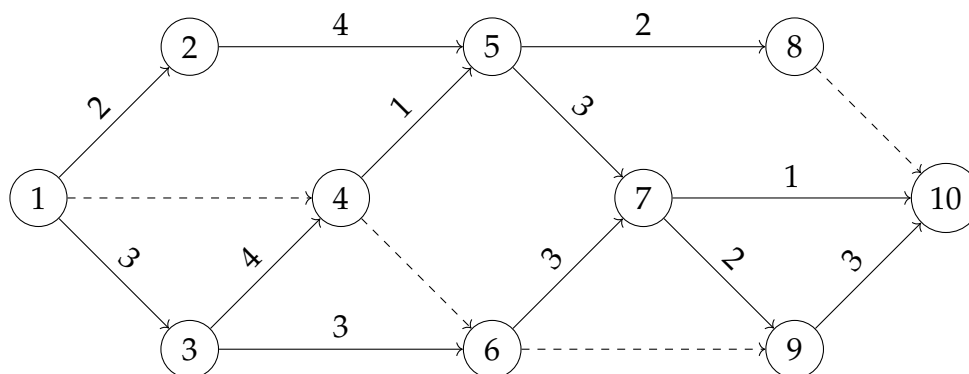
(e) Wie verläuft der kritische Pfad durch das Netzwerk?

Lösungsvorschlag

1 3 4 6 8 9

### 66115 / 2021 / Frühjahr / Thema 2 / Aufgabe 4

Gegeben ist das nachfolgende CPM-Netz. Gestrichelte Linien zwischen Ereignissen stellen Scheinvorgänge mit einer Dauer von 0 dar.



(a) Begründen Sie, welche Scheinvorgänge aus dem Netzplan ohne Informationsverlust gestrichen werden könnten.

Lösungsvorschlag

Die Scheinvorgänge zwischen den Ereignissen 1 und 4 bzw. zwischen 6 und 9 können jeweils gestrichen werden, da Ereignis 4 schon auf 1 wartet (über 3) und 9 wartet auf 6 (über 7).

(b) Berechnen Sie für jedes Ereignis den *frühesten Termin*, den *spätesten Termin* sowie die *Gesamtpufferzeiten*.

Lösungsvorschlag

$i$	Nebenrechnung	$FZ_i$
1		0
2		2
3		3
4		7
5	$\max(3_{(\rightarrow 3)} + 3, 7_{(\rightarrow 4)} + 1)$	8
6	$\max(3_{(\rightarrow 3)} + 3, 7_{(\rightarrow 4)} + 0)$	7
7	$\max(8_{(\rightarrow 5)} + 3, 7_{(\rightarrow 6)} + 3)$	11
8	$8_{(\rightarrow 5)} + 2$	10
9	$\max(7_{(\rightarrow 6)} + 0, 11_{(\rightarrow 7)} + 2)$	13
10	$\max(10_{(\rightarrow 7)} + 1, 8_{(\rightarrow 8)} + 0, 13_{(\rightarrow 9)} + 3)$	16

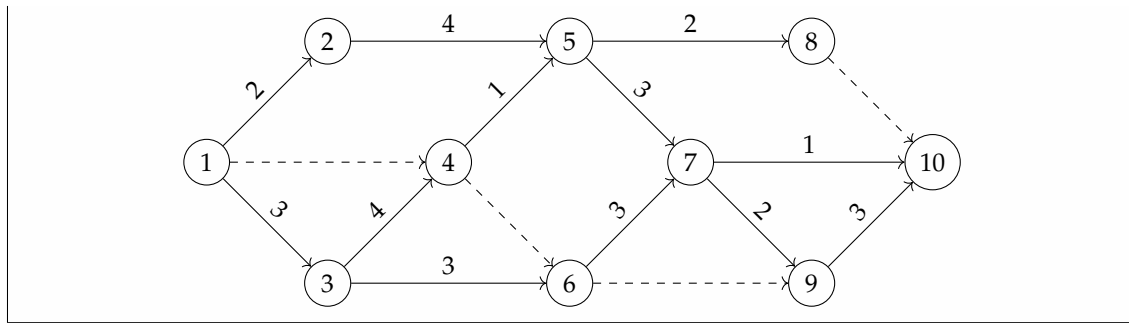
$i$	Nebenrechnung	$SZ_i$
1		0
2	$\min(8_{(\rightarrow 5)} - 4)$	4
3	$\min(8_{(\rightarrow 6)} - 3, 7_{(\rightarrow 4)} - 4)$	3
4	$\min(8_{(\rightarrow 5)} - 1, 8_{(\rightarrow 6)} - 0)$	7
5	$\min(16_{(\rightarrow 8)} - 2, 11_{(\rightarrow 7)} - 3)$	8
6	$\min(11_{(\rightarrow 7)} - 3, 13_{(\rightarrow 9)} - 0)$	8
7	$\min(16_{(\rightarrow 10)} - 1, 13_{(\rightarrow 9)} - 2)$	11
8	$16_{(\rightarrow 10)} - 0$	16
9	$16_{(\rightarrow 10)} - 3$	13
10	siehe $FZ_10$	16

$i$	1	2	3	4	5	6	7	8	9	10
$FZ_i$	0	2	3	7	8	7	11	10	13	16
$SZ_i$	0	4	3	7	8	8	11	16	13	16
GP	0	2	0	0	0	1	0	6	0	0

(c) Bestimmen Sie den kritischen Pfad.

Lösungsvorschlag

$1 \rightarrow 3 \rightarrow 4 \rightarrow 5 \rightarrow 7 \rightarrow 9 \rightarrow 10$



Gantt-Diagramm

## 66115 / 2021 / Frühjahr / Thema 2 / Aufgabe 4

In Gantt-Diagrammen unterscheidet man vier Anordnungsbeziehungen: Normalfolge (EA), Anfangsfolge (AA), Endfolge (EE) und Sprungfolge (AE). Ordnen Sie folgende Beispielen den Anordnungsbeziehungen (EA, AA, EE, AE) zu.

- (a) Tests durchführen und Dokumentation erstellen

Lösungsvorschlag

EE (Das Testen muss abgeschlossen sein bevor die Erstellung der Dokumentation beendet werden kann.)

- (b) Systementwurf und Implementierung

Lösungsvorschlag

AA (Der Beginn des Systementwurfs ist Voraussetzung für den Beginn der Implementierung.)

- (c) neue Anwendung in Betrieb nehmen und alte Anwendung abschalten

Lösungsvorschlag

AE (Die alte Anwendung kann erst abgeschaltet werden, wenn die neue Anwendung in Betrieb genommen wurde.)

- (d) Implementierung und Dokumentation

Lösungsvorschlag

EE (Die Implementierung muss abgeschlossen sein bevor die Dokumentation beendet werden kann.)

- (e) Abitur schreiben und Studieren

Lösungsvorschlag

EA (Das Abitur muss abgeschlossen sein bevor mit dem Studium begonnen werden kann.)

- (f) Führerschein machen und selbstständiges Autofahren

EA (Der Führerschein muss bestanden sein bevor mit dem selbstständigen Autofahren begonnen werden kann.)

- (g) Studieren und Buch aus Uni-Bibliothek ausleihen

AA (Der Beginn des Studiums ist Voraussetzung für das Ausleihen eines Buches aus der Uni-Bibliothek.)

## 66115 / 2021 / Frühjahr / Thema 2 / Aufgabe 4

Stellen Sie sich vor, Sie brauchen ein Grafiksystem. In diesem System wollen Sie aus Linien, Rechtecken, und Text größere Abbildungen darstellen. Diese Abbildungen sollen auch wieder andere Abbildungen rekursiv enthalten können. Sie brauchen also primitive Objekte: die Linie, das Rechteck und den Text. Zusätzlich brauchen Sie Behälter, die weitere Behälter und primitive Objekte enthalten können.

Erklären Sie, mit welchem Entwurfsmuster man diese Struktur abbilden kann und zeichnen Sie das dazugehörige Klassendiagramm.

## 66115 / 2021 / Frühjahr / Thema 2 / Aufgabe 4

- (a) Erklären Sie, was man bei der Entwicklung von Softwaresystemen unter einem Entwurfsmuster versteht und gehen Sie dabei auch auf die Vorteile ein.

Entwurfsmuster sind bewährte Lösungsschablonen für wiederkehrende Entwurfsprobleme sowohl in der Architektur als auch in der Softwarearchitektur und -entwicklung. Sie stellen damit eine wiederverwendbare Vorlage zur Problemlösung dar, die in einem bestimmten Zusammenhang einsetzbar ist.

Der primäre Nutzen eines Entwurfsmusters liegt in der Beschreibung einer Lösung für eine bestimmte Klasse von Entwurfsproblemen. Weiterer Nutzen ergibt sich aus der Tatsache, dass jedes Muster einen Namen hat. Dies vereinfacht die Diskussion unter Entwicklern, da man abstrakt über eine Struktur sprechen kann. So sind etwa Software-Entwurfsmuster zunächst einmal unabhängig von der konkreten Programmiersprache. Wenn der Einsatz von Entwurfsmustern dokumentiert wird, ergibt sich ein weiterer Nutzen dadurch, dass durch die Beschreibung des Musters ein Bezug zur dort vorhandenen Diskussion des Problemkontextes und der Vor- und Nachteile der Lösung hergestellt wird.

- (b) Nennen Sie die drei ursprünglichen Typen von Entwurfsmuster, erklären Sie diese kurz und geben Sie zu jedem Typ drei Beispiele an.

Typ	Erläuterung	Beispiele
Erzeugungsmuster	Dienen der Erzeugung von Objekten; diese wird dadurch gekapselt und ausgelagert, um den Kontext der Objekterzeugung unabhängig von der konkreten Implementierung zu halten	abstrakte Fabrik, Singleton, Prototyp
Strukturmuster	Erleichtern den Entwurf von Software durch vorgefertigte Schablonen für Beziehungen zwischen Klassen.	Adapter, Kompositum, Stellvertreter
Verhaltensmuster	Modellieren komplexes Verhalten der Software und erhöhen damit die Flexibilität der Software hinsichtlich ihres Verhaltens.	Beobachter, Interpreter, Iterator

## 66115 / 2021 / Frühjahr / Thema 2 / Aufgabe 4

- (a) Geben Sie zwei verschiedene Möglichkeiten der formalen Verifikation an.

Lösungsvorschlag

- 1. Möglichkeit:** formale Verifikation mittels *vollständiger Induktion* (eignet sich bei *rekursiven* Programmen).
- 2. Möglichkeit:** formale Verifikation mittels *wp-Kalkül* oder *Hoare-Kalkül* (eignet sich bei *iterativen* Programmen).

- (b) Erläutern Sie den Unterschied von partieller und totaler Korrektheit.

Lösungsvorschlag

- partielle Korrektheit:** Das Programm verhält sich spezifikationsgemäß, *falls* es terminiert.
- totale Korrektheit:** Das Programm verhält sich spezifikationsgemäß und es *terminiert immer*.

- (c) Gegeben sei die Anweisungssequenz  $A$ . Sei  $P$  die Vorbedingung und  $Q$  die Nachbedingung dieser Sequenz. Erläutern Sie, wie man die (partielle) Korrektheit dieses Programmes nachweisen kann.

Lösungsvorschlag

Vorgehen	Horare-Kalkül	wp-Kalkül
Wenn die Vorbedingung $P$ zutrifft, gilt nach der Ausführung der Anweisungssequenz $A$ die Nachbedingung $Q$ .	$\{P\}A\{Q\}$	$P \Rightarrow \text{wp}(A, Q)$

(d) Gegeben sei nun folgendes Programm:

```

A_1
while(b) :
  A_2
A_3

```

wobei  $A_1, A_2, A_3$  Anweisungssequenzen sind. Sei  $P$  die Vorbedingung und  $Q$  die Nachbedingung des Programms. Die Schleifeninvariante der while-Schleife wird mit  $I$  bezeichnet. Erläutern Sie, wie man die (partielle) Korrektheit dieses Programmes nachweisen kann.

Lösungsvorschlag

Vorgehen	Horare-Kalkül	wp-Kalkül
Die Invariante $I$ gilt vor Schleifeneintritt.	$\{P\}A_1\{I\}$	$P \Rightarrow \text{wp}(A_1, I)$
$I$ ist invariant, d. h. $I$ gilt nach jedem Schleifendurchlauf.	$\{I \wedge b\}A_2\{I\}$	$I \wedge b \Rightarrow \text{wp}(A_2, I)$
Die Nachbedingung $Q$ wird erfüllt.	$\{I \wedge \neg b\}A_3\{Q\}$	$I \wedge \neg b \Rightarrow \text{wp}(A_3, I)$

(e) Beschreiben Sie, welche Voraussetzungen eine Terminierungsfunktion erfüllen muss, damit die totale Korrektheit gezeigt werden kann.

Lösungsvorschlag

Mit einer Terminierungsfunktion  $T$  kann bewiesen werden, dass eine Wiederholung terminiert. Sie ist eine Funktion, die

- ganzzahlig,
  - nach unten beschränkt (die Schleifenbedingung ist *false*, wenn  $T = 0$ ) und
  - streng monoton fallend (jede Ausführung der Wiederholung verringert ihren Wert)
- ist.

Im Hoare-Kalkül muss  $\{I \wedge b \wedge (T = n)\}A\{T < n\}$  gezeigt werden, im wp-

Kalkül  $I \Rightarrow T \geq 0$ .<sup>a</sup>

wp-Kalkül

<sup>a</sup>[https://osg.informatik.tu-chemnitz.de/lehre/aup/aup-07-AlgorithmenEntwurf-script\\_de.pdf](https://osg.informatik.tu-chemnitz.de/lehre/aup/aup-07-AlgorithmenEntwurf-script_de.pdf)**66115 / 2021 / Frühjahr / Thema 2 / Aufgabe 4**

Gegeben sei folgendes Programm: wp-Kalkül

```

int f(int x, int y) {
    /* P */
    x = 2 * x + 1 + x * x;
    y += 7;
    if (x > 196) {
        y = 2 * y;
    } else {
        y -= 8;
        x *= 2;
    } /* Q */
    return x + y;
}

```

Bestimmen Sie die schwächste Vorbedingung (weakest precondition), für die die Nachbedingung  $Q := (x \geq 8) \wedge (y \% 2 = 1)$  noch zutrifft.

Lösungsvorschlag

Mit dem Distributivgesetz der Konjugation gilt:

$$\begin{aligned}
 \text{wp}("A; \text{if}(b) B; \text{else } C; ", Q) &\equiv \\
 \text{wp}("A; ", b) \wedge \text{wp}("A; B; ", Q) & \\
 \vee & \\
 \text{wp}("A; ", \neg b) \wedge \text{wp}("A; C; ", Q) &
 \end{aligned}$$

Der tatsächliche Programmcode wird eingesetzt:

$$\begin{aligned}
 \text{wp}("x=2*x+1+x*x; y+=7; \text{if}(x>196)\{y=2*y;\}\text{else}\{y-=8; x*=2;\}; ", (x \geq 8) \wedge (y \% 2 = 1)) &\equiv \\
 \text{wp}("x=2*x+1+x*x; y+=7; ", x > 196) \wedge & \\
 \text{wp}("x=2*x+1+x*x; y+=7; y=2*y; ", (x \geq 8) \wedge (y \% 2 = 1)) & \\
 \vee & \\
 \text{wp}("x=2*x+1+x*x; y+=7; ", x \leq 196) \wedge & \\
 \text{wp}("x=2*x+1+x*x; y+=7; y-=8; x*=2; ", (x \geq 8) \wedge (y \% 2 = 1)) & \\
 =: P &
 \end{aligned}$$

**Nebenrechnung:**  $\text{wp}("A; ", b)$

$\text{bWpPseudoMatheUmgebung } \text{wp}("x=2*x+1+x*x; y+=7; ", x > 196)$

Wir lassen  $y += 7$  weg, weil in der Nachbedingung kein  $y$  vorkommt und setzen in den Term  $x > 196$  für das  $x$  die erste Code-Zeile  $2 \cdot x + 1 + x \cdot x$  ein.

$$\equiv \text{wp}("", 2 \cdot x + 1 + x \cdot x > 196)$$

Nach der Transformationsregel *Nichts passiert, die Vorbedingung bleibt gleich* kann das auch so geschrieben werden:



$$\equiv 2 \cdot x + 1 + x \cdot x > 196$$

Die erste binomische Formel (Plus-Formel) lautet  $(a + b)^2 = a^2 + 2ab + b^2$ . Man kann die Formel auch umgedreht verwenden:  $a^2 + 2ab + b^2 = (a + b)^2$ . Die erste Code-Zeile  $2 \cdot x + 1 + x \cdot x$  kann umformuliert werden in  $1 + 2 \cdot 1 \cdot x + x \cdot x = 1^2 + 2 \cdot 1 \cdot x + x^2 = (1 + x)^2 = (x + 1)^2$ . Wir haben für  $a$  die Zahl 1 und für  $b$  den Buchstaben  $x$  eingesetzt.

$$\equiv (x + 1)^2 > 196$$

**Nebenrechnung: wp("A;B;", Q)**

**bWpPseudoMatheUmgebung wp("x=2\*x+1+x\*x;y+=7;y=2\*y;", (x ≥ 8) ∧ (y%2 = 1))**

Für das  $x$  in der Nachbedingung setzen wir die erste Code-Zeile  $2 \cdot x + 1 + x \cdot x$  ein. Für das  $y$  in der Nachbedingung setzen wir dritte Code-Zeile  $y=2*y$ ; ein und dann die zweite Code-Zeile  $y+=7$ ; . Das wp-Kalkül arbeitet den Code rückwärts ab. in  $y\%2$  die dritte Anweisung  $y = 2 \cdot y$  einfügen:  $2 \cdot y\%2$  dann in  $2 \cdot y\%2$  die zweite Anweisung  $y = y + 7$  einfügen:  $2 \cdot (y + 7)\%2$

$$\equiv (x + 1)^2 \geq 8 \wedge 2(y + 7)\%2 = 1$$

Diese Aussage ist falsch, da  $2(y + 7)$  immer eine gerade Zahl ergibt und der Rest von einer Division durch zwei einer geraden Zahl immer 0 ist und nicht 1.

$$\equiv (x + 1)^2 \geq 8 \wedge \text{falsch}$$

$$\equiv \text{falsch}$$

**Nebenrechnung: wp("A;", ¬b)**

**bWpPseudoMatheUmgebung wp("x=2\*x+1+x\*x;y+=7;", x ≤ 196)**

Analog zu Nebenrechnung 1

$$\equiv (x + 1)^2 \leq 196$$

**Nebenrechnung: wp("A;c;", Q)**

**bWpPseudoMatheUmgebung wp("x=2\*x+1+x\*x;y+=7;y-=8;x\*=2;", (x ≥ 8) ∧ (y%2 = 1))**

„x\*=2;“:  $x \cdot 2$  für  $x$  einsetzen:

$$\equiv \text{wp}("x=2*x+1+x*x;y+=7;y-=8;", (2 \cdot x \geq 8) \wedge (y\%2 = 1))$$

„y-=8;“:  $y - 8$  für  $y$  einsetzen:

$$\equiv \text{wp}("x=2*x+1+x*x;y+=7;", (2 \cdot x \geq 8) \wedge ((y - 8)\%2 = 1))$$

„y+=7“:  $y + 7$  für  $y$  einsetzen:

$$\equiv \text{wp}("x=2*x+1+x*x;", (2 \cdot x \geq 8) \wedge (((y + 7) - 8)\%2 = 1))$$

„x=2\*x+1+x\*x;“:  $(x + 1)^2$  für  $x$  einsetzen:

$$\equiv \text{wp}("", (2 \cdot (x + 1)^2 \geq 8) \wedge (((y + 7) - 8)\%2 = 1))$$

Nur noch die Nachbedingung stehen lassen:

$$\equiv (2 \cdot (x + 1)^2 \geq 8) \wedge (((y + 7) - 8)\%2 = 1)$$

Subtraktion:

$$\equiv (2 \cdot (x+1)^2 \geq 8) \wedge ((y-1)\%2 = 1)$$

Vereinfachen (links beide Seiten durch 2 teilen und rechts von beiden Seiten 1 abziehen)

$$\equiv \left(\frac{2 \cdot (x+1)^2}{2} \geq \frac{8}{2}\right) \wedge (((y-1)\%2) - 1 = 1 - 1)$$

Zwischenergebnis:

$$\equiv ((x+1)^2 \geq 4) \wedge y\%2 = 0$$

## Zusammenführung

Die Zwischenergebnisse aus den Nebenrechnungen zusammenfügen:

$$\equiv [(x+1)^2 > 196 \wedge \text{falsch}] \vee [(x+1)^2 \leq 196 \wedge (x+1)^2 \geq 4 \wedge y\%2 = 0]$$

„falsch“ und eine Aussage verbunden mit logischem Und „ $\wedge$ “ ist insgesamt falsch:

$$\equiv \text{falsch} \vee [(x+1)^2 \leq 196 \wedge (x+1)^2 \geq 4 \wedge y\%2 = 0]$$

falsch verbunden mit oder weglassen:

$$\equiv (x+1)^2 \leq 196 \wedge (x+1)^2 \geq 4 \wedge y\%2 = 0$$

Umgruppieren, sodass nur noch ein  $(x+1)^2$  geschrieben werden muss:

$$\equiv 4 \leq (x+1)^2 \leq 196 \wedge y\%2 = 0$$

$$4 = 2^2 \text{ und } 196 = 14^2$$

$$\equiv 2^2 \leq (x+1)^2 \leq 14^2 \wedge y\%2 = 0$$

Hoch zwei weg lassen: Betragsklammer  $|x|$  oder auch Betragsfunktion hinzufügen (Die Betragsfunktion ist festgelegt als „Abstand einer Zahl von der Zahl Null“.

$$\equiv 2 \leq |x+1| \leq 14 \wedge y\%2 = 0$$

Auf die Gleichung der linken Aussage  $-1$  anwenden:

$$\equiv 1 \leq |x| \leq 13 \wedge y\%2 = 0$$

Die Betragsklammer weg lassen:

$$\equiv (1 \leq x \leq 13 \vee -13 \leq x \leq -1) \wedge y\%2 = 0$$

bWpPseudoMatheUmgebung=: P

## 66115 / 2021 / Frühjahr / Thema 2 / Aufgabe 4

Bestimmen Sie zur Nachbedingung  $Q$  die Vorbedingung  $P$ !

**Nachbedingung:**  $Q \equiv x + y = 17$

**Programmcode:**

```
// P: ?
x += 5;
y *= 2;
z = z % 4;
y--;
// Q: x + y = 17
```

Lösungsvorschlag

ist gleichbedeutend mit

```
x = x + 5;
y = y * 2;
z = z % 4;
y = y - 1;
```

$\text{wp}("x += 5; y *= 2; z = z \% 4; y--;", x + y = 17)$

$y - 1$  einsetzen

$\equiv \text{wp}("x += 5; y *= 2; z = z \% 4;", x + y - 1 = 17)$

die 1 mit + nach rechts bringen

$\equiv \text{wp}("x += 5; y *= 2; z = z \% 4;", x + y = 18)$

Im nächsten Schritt müssten wir ein  $z$  verändern. Wir haben aber in unserer Bedingung kein  $z$ , deshalb kann es wegfallen.

$\equiv \text{wp}("x += 5; y *= 2;", x + y = 18)$

$y \cdot 2$  einsetzen

$\equiv \text{wp}("x += 5;", x + y \cdot 2 = 18)$

Auf  $x$  wird 5 hinzuaddiert.

$\equiv \text{wp}("", x + 5 + y \cdot 2 = 18)$

Wir haben keinen Programmcode mehr. Wir können  $\text{wp}$  weglassen.

$\equiv x + 5 + y \cdot 2 = 18$

Die 5 nach rechts bringen

$\equiv x + y \cdot 2 = 13$

Alle Eingaben die Vorbedingung  $P \equiv x + y \cdot 2 = 13$  erfüllen, erfüllen die Nachbedingung  $Q \equiv x + y = 17$ .

## 66115 / 2021 / Frühjahr / Thema 2 / Aufgabe 4

Gegeben sei folgende Methode:

```
public int ggT(int a, int b) {
    int result = 1;
    for (int i = 1; i <= Math.min(a, b); i++) {
        if ((a % i == 0) & (b % i == 0)) {
            result = i;
        }
    }
}
```

```

}
return result;
}

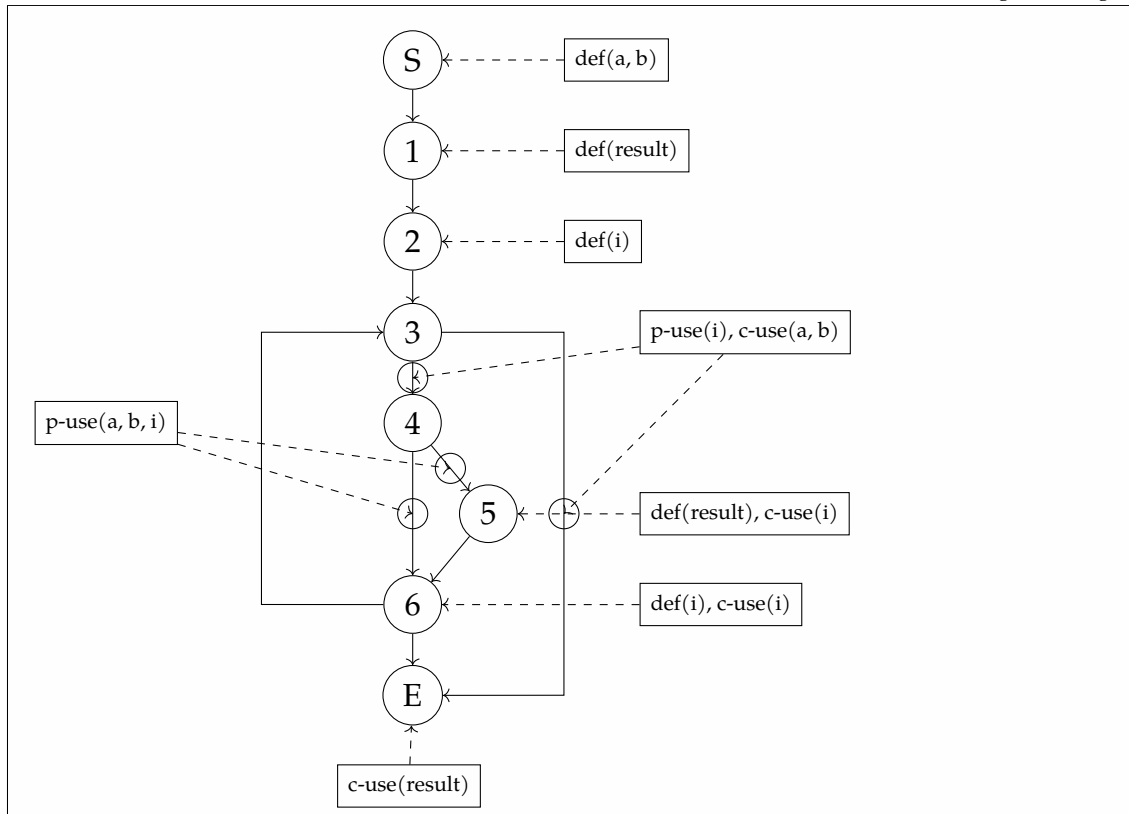
```

Datenfluss-annotierter  
Kontrollflussgraph  
Zyklomatische Komplexität  
nach McCabe

Code-Beispiel auf Github ansehen: [src/main/java/org/bschlangaul/aufgaben/sosy/white\\_box/WhiteBox.java](https://github.com/src/main/java/org/bschlangaul/aufgaben/sosy/white_box/WhiteBox.java)

- (a) Erstellen Sie den zur Methode gehörenden datenflussannotierten Kontrollflussgraphen.

Lösungsvorschlag



- (b) Geben Sie die zyklomatische Komplexität  $M$  nach McCabe der Methode `ggT` an. (Nur das Ergebnis!)

Lösungsvorschlag

Berechnung durch Anzahl Binärverzweigungen  $b$  ( $p$  Anzahl der Zusammenhangskomponenten des Kontrollflussgraphen)

$$M = b + p$$

$$\rightarrow M = 2 + 1 = 3$$

oder durch Anzahl Kanten  $e$  und Knoten  $n$

$$M = e - n + 2p$$

$$\rightarrow M = 9 - 8 + 2 \cdot 1 = 3$$

- (c) Geben Sie je einen Repräsentanten aller Pfadklassen im Kontrollflussgraphen an, die zum Erzielen einer vollständigen Schleifen-Inneres-Überdeckung (Boundary-Interior-Coverage) genügen würden.

Lösungsvorschlag

**Äußere Pfade**

- S 1 2 3 E

**Grenzpfade**

- S 1 2 3 4 5 6 3 E

- S 1 2 3 4 6 3 E

**Innere Pfade**

- S 1 2 3 4 5 6 3 4 5 6 3 E

- S 1 2 3 4 6 3 4 6 3 E

- S 1 2 3 4 5 6 3 4 6 3 E

- S 1 2 3 4 6 3 4 5 6 3 E

- (d) Geben Sie an, welche der Pfade aus der vorherigen Aufgabe nicht überdeckbar ("feasible") sind und begründen Sie dies.

Lösungsvorschlag

**Äußere Pfade**

S 1 2 3 E ja, z. B.  $\text{ggT}(-1, -2)$ .

**Grenzpfade**

S 1 2 3 4 5 6 3 E ja, z. B.  $\text{ggT}(10, 20)$ .

S 1 2 3 4 6 3 E ja, z. B.  $\text{ggT}(1, 2)$ .

**Innere Pfade**

S 1 2 3 4 5 6 3 4 5 6 3 E ja, z. B.  $\text{ggT}(2, 2)$ .

S 1 2 3 4 6 3 4 6 3 E nicht feasible, da geteilt durch eins immer Modulo 0 ergibt, egal welche Zahl a oder b hat. Bei der ersten Schleifenwiederholung wird immer die innere If-Verzweigung genommen.

S 1 2 3 4 5 6 3 4 6 3 E ja, z. B.  $\text{ggT}(2, 3)$ .

**S 1 2 3 4 6 3 4 5 6 3 E** nicht feasible, da geteilt durch eins immer Modulo 0 ergibt, egal welche Zahl a oder b hat. Bei der ersten Schleifenwiederholung wird immer die innere If-Verzweigung genommen.

## 66115 / 2021 / Frühjahr / Thema 2 / Aufgabe 4

- (a) Nennen Sie die wesentlichen Unterschiede zwischen *White-Box-Testen* und *Black-Box-Testen* und geben Sie jeweils zwei Beispiele an.

Lösungsvorschlag

**White-Box-Tests** sind *strukturorientiert*, z. B. Kontrollflussorientiertes Testen oder Datenflussorientiertes Testen.

**Black-Box-Test** sind *spezifikations- und funktionsorientiert* (nur Ein- und Ausgabe relevant), z. B. Äquivalenzklassenbildung oder Grenzwertanalyse.

- (b) Geben Sie drei nicht-funktionalorientierte Testarten an.

Lösungsvorschlag

- (i) Performanztest
- (ii) Lasttest
- (iii) Stresstest

- (c) Nennen Sie die vier verschiedenen Teststufen aus dem V-Modell und erläutern Sie deren Ziele.

Lösungsvorschlag

**Komponenten-Test:** Fehlerzustände in Modulen finden.

**Integrations-Test:** Fehlerzustände in Schnittstellen und Interaktionen finden.

**System-Test:** Abgleich mit Spezifikation.

**Abnahme-Test:** Vertrauen in System und nicht-funktionale Eigenschaften gewinnen.

- (d) Nennen Sie fünf Aktivitäten des Testprozesses.

Lösungsvorschlag

- (i) Testplanung und Steuerung,
- (ii) Testanalyse und Testentwurf,
- (iii) Testrealisierung und Testdurchführung,
- (iv) Bewertung von Endkriterien und Bericht,
- (v) Abschluss der Testaktivitäten

## 66115 / 2021 / Frühjahr / Thema 2 / Aufgabe 4

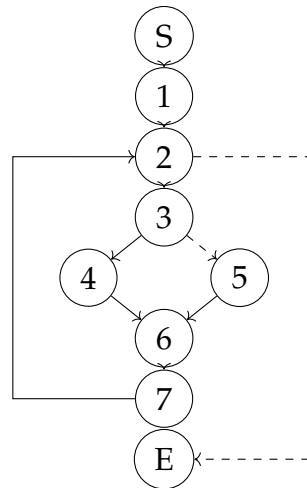
Gegeben sei folgende Methode und ihr Kontrollflussgraph:

```

int log(int a) {
    int x = a;
    int z = 0;
    while (x > 1) {
        if (x % 2 == 0) {
            z++;
            x /= 2;
        } else {
            x--;
        }
    }
    return z;
}

```

Code-Beispiel auf Github ansehen:  
[src/main/java/org/bschlangaul/aufgaben/sosy/ab\\_7/Aufgabe3.java](https://github.com/bschlangaul/aufgaben/sosy/ab_7/Aufgabe3.java)

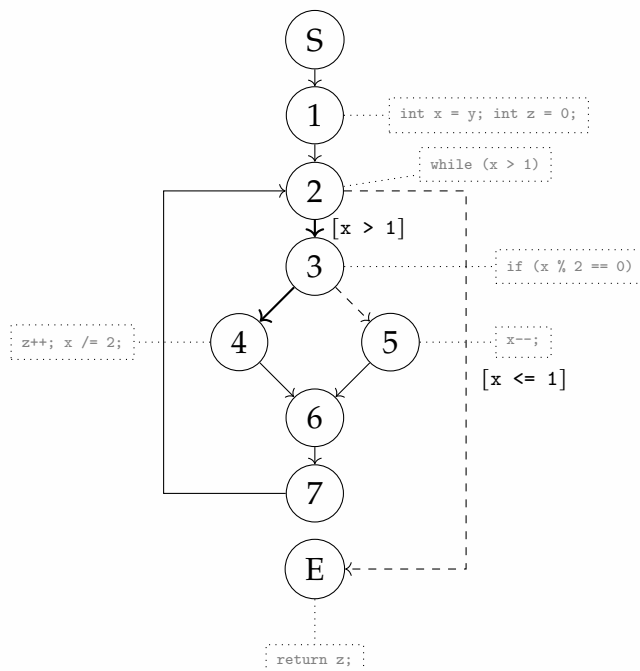


Überdeckbarkeit  
 C2b Schleife-Inneres-  
 Pfadüberdeckung  
 (Boundary-Interior Path  
 Coverage)

- (a) Begründen Sie, warum der Pfad  $S - 1 - 2 - 3 - 5 - 6 - 7 - 2 - E$  infeasible (= nicht überdeckbar) ist, also weshalb es keine Eingabe gibt, unter der dieser Pfad durchlaufen werden kann.

Lösungsvorschlag

Damit dieser Pfad durchlaufen werden könnte, müsste die Eingabe  $a$  gleichzeitig 2 und ungerade sein.



- (b) Geben Sie eine minimale Menge von Pfaden an, mit der eine vollständigen Schleifen-Inneres-Überdeckung erzielt werden kann, sowie gegebenenfalls zu jedem Pfad eine Eingabe, unter der dieser Pfad durchlaufen werden kann.

Lösungsvorschlag

**ohne Schleifenausführung**

-  $S - 1 - 2 - E$  ( $a=1$ )

**boundary-Tests**

- (S) - (1) - (2) - (3) - (4) - (6) - (7) - (2) - (E)  
(a=2)

- (S) - (1) - (2) - (3) - (5) - (6) - (7) - (2) - (E)  
(infeasible)

**interior-Tests**

- (S) - (1) - (2) - (3) - (4) - (6) - (7) - (2) - (3) - (4) - (6) - (7) - (2) - (E)  
(a=4)

- (S) - (1) - (2) - (3) - (4) - (6) - (7) - (2) - (3) - (5) - (6) - (7) - (2) - (E)  
(a=6)

- (S) - (1) - (2) - (3) - (5) - (6) - (7) - (2) - (3) - (4) - (6) - (7) - (2) - (E)  
(a=3)

- (S) - (1) - (2) - (3) - (5) - (6) - (7) - (2) - (3) - (5) - (6) - (7) - (2) - (E)  
(infeasible)

**46115 / 2015 / Herbst / Thema 2 / Aufgabe 4**

Gegeben sei die folgende Methode `function`:

```
double function(int n) {
    if (n == 1)
        return 0.5 * n;
    else
        return 1.0 / (n * (n + 1)) + function(n - 1);
}
```

Code-Beispiel auf Github ansehen: [src/main/java/org/bschlangaul/examen/examen\\_46115/jahr\\_2015/herbst/Induktion.java](https://github.com/src/main/java/org/bschlangaul/examen/examen_46115/jahr_2015/herbst/Induktion.java)

Beweisen Sie folgenden Zusammenhang mittels vollständiger Induktion:

$$\forall n \geq 1: \text{function}(n) = f(n) \text{ mit } f(n) := 1 - \frac{1}{n+1}$$

Hinweis: Eventuelle Rechenungenauigkeiten, wie z. B. in Java, bei der Behandlung von Fließkommazahlen (z. B. `double`) sollen beim Beweis nicht berücksichtigt werden - Sie dürfen also annehmen, Fließkommazahlen würden mathematische Genauigkeit aufweisen.

Lösungsvorschlag

**Induktionsanfang**

— Beweise, dass  $A(1)$  eine wahre Aussage ist. \_\_\_\_\_

$$f(1) := 1 - \frac{1}{1+1} = 1 - \frac{1}{2} = \frac{1}{2}$$

**Induktionsvoraussetzung**

— Die Aussage  $A(k)$  ist wahr für ein beliebiges  $k \in \mathbb{N}$ . \_\_\_\_\_

$$f(n) := 1 - \frac{1}{n+1}$$



**Induktionsschritt**

— Beweise, dass wenn  $A(n = k)$  wahr ist, auch  $A(n = k + 1)$  wahr sein muss. —

**zu zeigen:**

$$f(n + 1) := 1 - \frac{1}{(n+1)+1} = f(n)$$

**Vorarbeiten (Java in Mathe umwandeln):**

$$\text{function}(n) = \frac{1}{n \cdot (n+1)} + f(n - 1)$$

$$\begin{aligned}
 f(n + 1) &= \frac{1}{(n + 1) \cdot ((n + 1) + 1)} + f((n + 1) - 1) && n + 1 \text{ eingesetzt} \\
 &= \frac{1}{(n + 1) \cdot (n + 2)} + f(n) && \text{vereinfacht} \\
 &= \frac{1}{(n + 1) \cdot (n + 2)} + 1 - \frac{1}{n + 1} && \text{für } f(n) \text{ Formel eingesetzt} \\
 &= 1 + \frac{1}{(n + 1) \cdot (n + 2)} - \frac{1}{n + 1} && \text{1. Bruch an 2. Stelle geschrieben} \\
 &= 1 + \frac{1}{(n + 1) \cdot (n + 2)} - \frac{1 \cdot (n + 2)}{(n + 1) \cdot (n + 2)} && \text{2. Bruch mit } (n + 2) \text{ erweitert} \\
 &= 1 + \frac{1 - (n + 2)}{(n + 1) \cdot (n + 2)} && \text{die 2 Brüche subtrahiert} \\
 &= 1 + \frac{1 - n - 2}{(n + 1) \cdot (n + 2)} && - + 2 = -2 \\
 &= 1 + \frac{-1 - n}{(n + 1) \cdot (n + 2)} && 1 - 2 = -1 \\
 &= 1 + \frac{-1 \cdot (1 + n)}{(n + 1) \cdot (n + 2)} && (n + 1) \text{ ausgeklammert} \\
 &= 1 + \left( -1 \cdot \frac{(1 + n)}{(n + 1) \cdot (n + 2)} \right) && \text{minus vor den Bruch bringen} \\
 &= 1 - \frac{(1 + n)}{(n + 1) \cdot (n + 2)} && \text{plus minus ist minus} \\
 &= 1 - \frac{1}{n + 2} && (n + 1) \text{ gekürzt} \\
 &= 1 - \frac{1}{(n + 1) + 1} && \text{Umformen zur Verdeutlichung}
 \end{aligned}$$

**46116 / 2010 / Frühjahr / Thema 1 / Aufgabe 1**

Es sei folgender Sachverhalt gegeben:

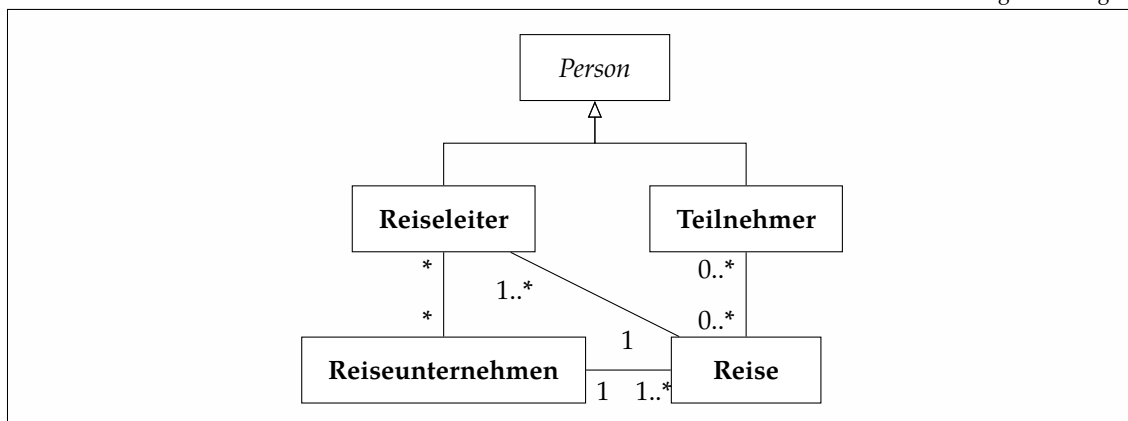
Ein Reiseunternehmen bietet verschiedene Reisen an. Dazu beschäftigt es eine Reihe

von Reiseleitern, wobei eine Reise von mindestens einem Reiseleiter geleitet wird. Da Reiseleiter freiberuflich arbeiten, können sie bei mehreren Reiseunternehmen Reisen leiten.

An einer Reise können mehrere Teilnehmer teilnehmen, ein Teilnehmer kann auch an verschiedenen Reisen teilnehmen.

- (a) Modellieren Sie diesen Sachverhalt in einem UML-Klassendiagramm. Für Teilnehmer und Reiseleiter sollen Sie dabei eine abstrakte Oberklasse definieren. Achten Sie dabei auf die Multiplizitäten der Assoziationen. Sie müssen keine Attribute bzw. Methoden angeben.

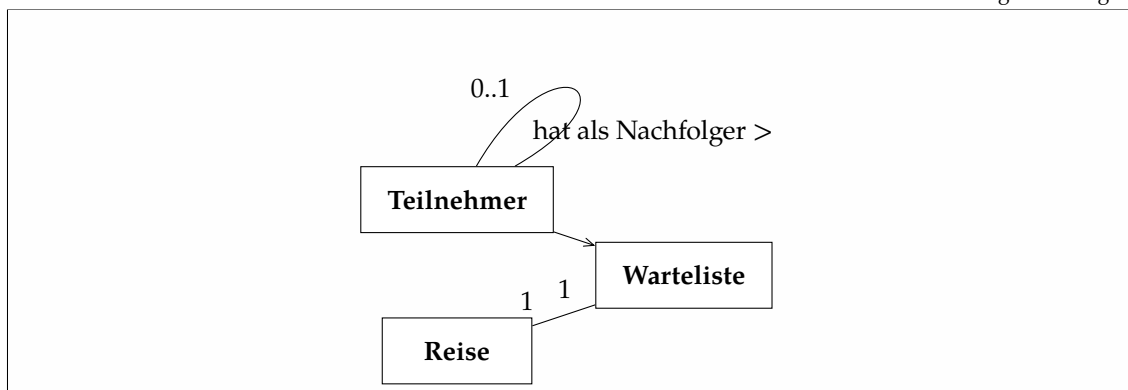
Lösungsvorschlag



- (b) Eine Reise kann jedoch nur mit einer begrenzten Kapazität angeboten werden, das heißt, zu einer bestimmten Reise kann nur eine begrenzte Anzahl von Teilnehmern assoziiert werden. Als Ausgleich soll pro Reise eine Warteliste verwaltet werden.

Modellieren Sie diesen erweiterten Sachverhalt in einem neuen Diagramm. Nicht veränderte Klassen brauchen nicht noch einmal angegeben werden. Beachten Sie dabei, dass die Reihenfolge bei einer Warteliste eine Rolle spielt.

Lösungsvorschlag



- (c) Implementieren Sie die in Aufgabenteil b) modellierten Klassen in Java. Fügen Sie eine Methode hinzu, die einen Teilnehmer von einer Reise entfernt. Dabei soll automatisch der erste Platz der Warteliste zu einem ReisetTeilnehmer werden, wenn

die Warteliste nicht leer ist. Achten Sie auf die Navigierbarkeit Ihrer Assoziationen. Sie können davon ausgehen, dass die Methode nur mit Teilnehmern aufgerufen wird, die in der Tat Teilnehmer der Reise sind.

```
public class Teilnehmer {  
    Teilnehmer nächster;  
  
}
```

Code-Beispiel auf Github ansehen: [src/main/java/org/bschlangaul/examen/examen\\_46116/jahr\\_2010/fruehjahr/reiseunternehmen/Teilnehmer.java](#)

```
/**  
 * Diese Klasse ist eine Implementation einer einfach verketteten Liste. Sie  
 * wird einerseits in der Klasse {@link Reise} genutzt, um die Reiseteilnehmer zu  
 * speichern, andererseits um eine Warteliste darauf aufbauen zu können.  
 */  
public class TeilnehmerListe {  
  
}
```

Code-Beispiel auf Github ansehen: [src/main/java/org/bschlangaul/examen/examen\\_46116/jahr\\_2010/fruehjahr/reiseunternehmen/TeilnehmerListe.java](#)

```
/**  
 * Eine Reise kann jedoch nur mit einer begrenzten Kapazität angeboten  
 * werden, das heißt, zu einer bestimmten Reise kann nur eine begrenzte  
 * Anzahl von Teilnehmern assoziiert werden. Als Ausgleich soll pro  
 * Reise eine Warteliste verwaltet werden.  
 *  
 * Modellieren Sie diesen erweiterten Sachverhalt in einem neuen  
 * Diagramm. Nicht veränderte Klassen brauchen nicht noch einmal  
 * angegeben werden. Beachten Sie dabei, dass die Reihenfolge bei einer  
 * Warteliste eine Rolle spielt.  
 *  
 * Implementieren Sie die in Aufgabenteil b) modellierten Klassen in  
 * Java. Fügen Sie eine Methode hinzu, die einen Teilnehmer von einer  
 * Reise entfernt. Dabei soll automatisch der erste Platz der Warteliste  
 * zu einem Reiseteilnehmer werden, wenn die Warteliste nicht leer ist.  
 * Achten Sie auf die Navigierbarkeit Ihrer Assoziationen. Sie können  
 * davon ausgehen, dass die Methode nur mit Teilnehmern aufgerufen wird,  
 * die in der Tat Teilnehmer der Reise sind.  
 */  
public class Warteliste extends TeilnehmerListe {  
  
}
```

Code-Beispiel auf Github ansehen: [src/main/java/org/bschlangaul/examen/examen\\_46116/jahr\\_2010/fruehjahr/reiseunternehmen/Warteliste.java](#)

```
public class Reise {  
  
    Teilnehmer teilnehmer;  
  
    Warteliste warteliste;
```

```

void entferneTeilnehmer(Teilnehmer teilnehmer) {

}

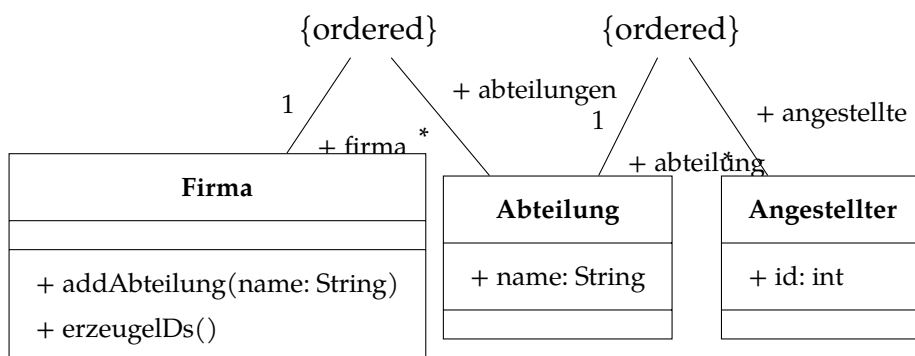
}

```

Code-Beispiel auf Github ansehen: [src/main/java/org/bschlangaul/examen/examen\\_46116/jahr\\_2010/fruehjahr/reiseunternehmen/Reise.java](https://github.com/bschlangaul/examen/examen_46116/jahr_2010/fruehjahr/reiseunternehmen/Reise.java)

## 46116 / 2011 / Frühjahr / Thema 1 / Teilaufgabe 2 / Aufgabe 1

### Firmenstruktur

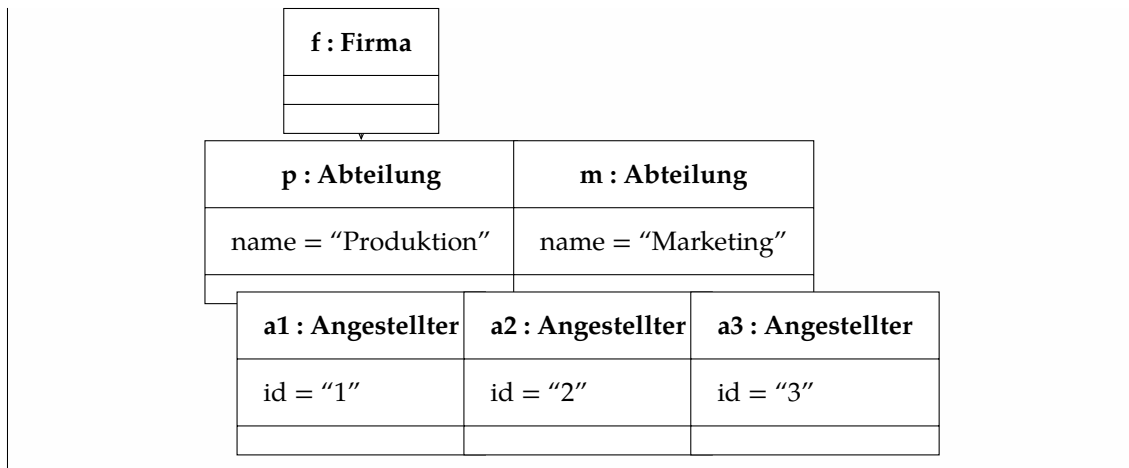


Eine Firma besteht aus `null` oder mehr Abteilungen, von denen jede `null` oder mehr Angestellte hat. Da sowohl die Abteilungen als auch deren Angestellten geordnet sind, sind Angestellte insgesamt geordnet. Sie haben durchgehende, ganzzahlige IDs, die bei 1 beginnen.

- (a) Erstellen Sie exemplarisch ein Objektdiagramm: Stellen Sie eine Firma mit dem Instanznamen `f` und den zwei Abteilungen „Produktion“ (Name `p`) und „Marketing“ (Name `m`) dar. Die Produktion hat zwei Angestellte, Marketing hat einen Angestellten. Die Angestellten haben die Namen `a1`, `a2`, und `a3`.

Lösungsvorschlag

Die Instanzbezeichnung müsste noch unterstrichen werden. Das geht aber leider mit TikZ-UML nicht.



- (b) Implementieren Sie das Klassendiagramm in Java oder in einer anderen geeigneten objektorientierten Programmiersprache Ihrer Wahl. Beachten Sie, dass die Assoziationen bidirektional und geordnet sind. Die beiden Methoden der Klasse **Firma** sollen dabei folgendes Verhalten haben:

Die Methode **erzeugeIDs** sorgt dafür, dass die IDs wieder korrekt zugewiesen sind. Die alten IDs können beliebig geändert werden, solange das Endergebnis wieder den obenstehenden Kriterien genügt.

Lösungsvorschlag

```
import java.util.ArrayList;
import java.util.List;

public class Firma {

    List<Abteilung> abteilungen;

    public Firma() {
        abteilungen = new ArrayList<Abteilung>();
    }

    public void addAbteilung(String name) {
        for (Abteilung abteilung : abteilungen) {
            if (abteilung.name.equals(name)) {
                → System.out.println("Eine Abteilung mit diesem Namen gibt es bereits schon.");
                return;
            }
            abteilungen.add(new Abteilung(name));
        }
    }

    public void erzeugeIDs() {
        int idZähler = 1;
        for (Abteilung abteilung : abteilungen) {
            for (Angestellter angestellter : abteilung.angestellte) {
                angestellter.id = idZähler;
                idZähler++;
            }
        }
    }
}
```

```
}  
}  
}
```

Code-Beispiel auf Github ansehen: [src/main/java/org/bschlangaul/examen/examen\\_46116/jahr\\_2011/fruehjahr/Firma.java](#)

```
import java.util.ArrayList;  
import java.util.List;  
  
public class Abteilung {  
    public String name;  
  
    public List<Angestellter> angestellte;  
  
    public Abteilung(String name) {  
        this.name = name;  
        angestellte = new ArrayList<Angestellter>();  
    }  
}
```

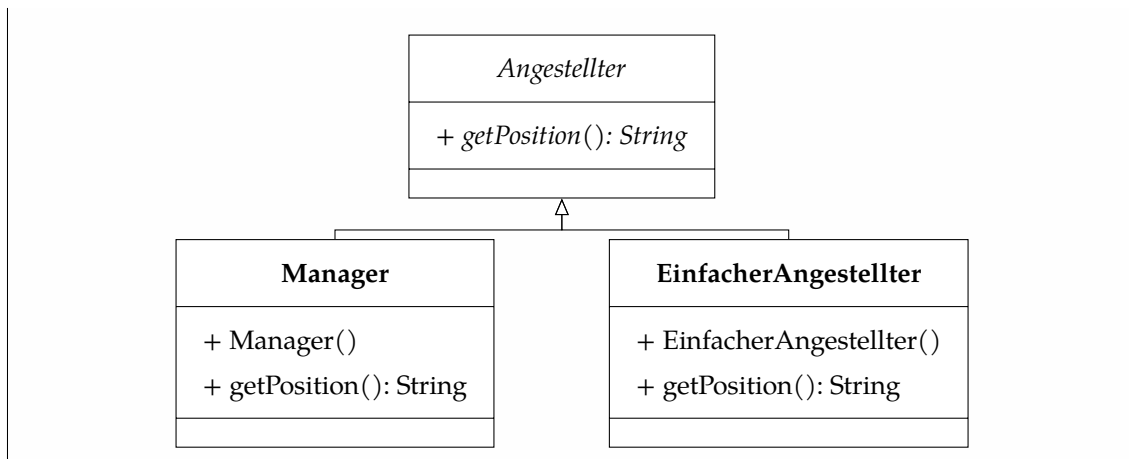
Code-Beispiel auf Github ansehen: [src/main/java/org/bschlangaul/examen/examen\\_46116/jahr\\_2011/fruehjahr/Abteilung.java](#)

```
import java.util.List;  
  
public class Angestellter {  
    public int id;  
  
    public Firma firma;  
  
    public List<Angestellter> angestellte;  
}
```

Code-Beispiel auf Github ansehen: [src/main/java/org/bschlangaul/examen/examen\\_46116/jahr\\_2011/fruehjahr/Angestellter.java](#)

- (c) Angestellte sollen in Manager und einfache Angestellte unterteilt werden. Zeichnen Sie ein Klassendiagramm mit der Oberklasse `Angestellter` und den zwei Unterklassen `Manager` und `EinfacherAngestellter`. Die Klasse `Angestellter` soll nicht instantiierbar sein und erzwingen, dass die Methode `getPosition()` (öffentlich, ohne Argumente, Rückgabewert `String`) von allen konkreten Unterklassen implementiert wird. `Manager` und `EinfacherAngestellter` sollen instantiierbar sein.

Lösungsvorschlag



- (d) Wie lautet der Fachbegriff dafür, dass eine Methode in einer Klasse und in deren Unterklassen dieselbe Signatur hat, aber in den Unterklassen unterschiedlich implementiert ist?

Lösungsvorschlag

Abstrakte Methode

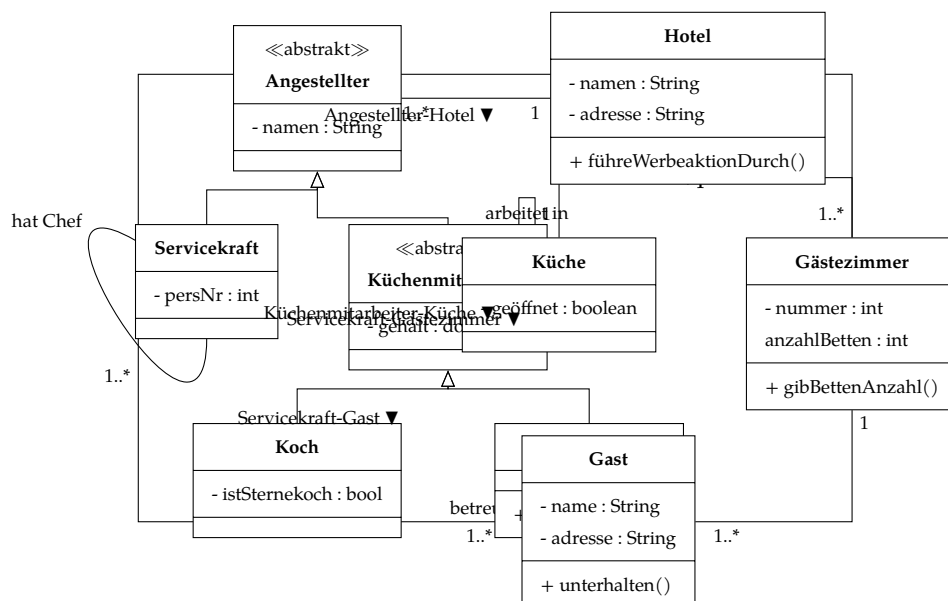
## 46116 / 2012 / Frühjahr / Thema 2 / Teilaufgabe 2 / Aufgabe 1

### Hotel-Verwaltung

Erstellen Sie zu der folgenden Beschreibung eines Systems zur Organisation eines Hotels ein Klassendiagramm, das Attribute und Assoziationen mit Kardinalitäten sowie Methoden enthält. Setzen Sie dabei das Konzept der Vererbung sinnvoll ein.

- Ein **Hotel** besteht aus genau einer Küche und mehreren Gästezimmern.
- Es hat einen Namen, eine Adresse und kann Werbeaktionen durchführen.
- Alle Mitarbeiter, sowohl Servicekräfte als auch Küchenmitarbeiter, sind Angestellte des Hotels.
- Die Küche kann geöffnet oder geschlossen sein.
- Gästezimmer haben eine Nummer und eine bestimmte Anzahl an Betten, die ausgegeben werden kann.
- In diesen Gästezimmern wohnen Gäste, die von einer oder mehreren Servicekräften umsorgt werden.
- Servicekräfte sind nur für die ihnen zugeordneten Gästezimmer verantwortlich.
- Jede Servicekraft hat einen Namen und eine Personalnummer sowie einen Vorgesetzten, der auch eine Servicekraft ist.

- In der Küche arbeiten Küchenmitarbeiter, die einen Namen haben und ein Gehalt bekommen.
- Küchenmitarbeiter sind entweder Köche oder Aushilfen. Köche können zudem Sterneköche sein, also mit einem oder mehreren Sternen dekoriert sein.
- Aushilfen bauen die Buffets für die Mahlzeiten auf.
- Gäste können sich unterhalten. Jeder Gast hat einen Namen und eine Adresse und ist seinem Zimmer zugeordnet.



## 46116 / 2013 / Frühjahr / Thema 1 / Teilaufgabe 1 / Aufgabe 2

Erläutern Sie in jeweils ca. 3 Sätzen die folgenden Begriffe aus der modernen Software-technologie: *Pair Programming*, *Refactoring*, *agile Softwareentwicklung*.



**Pair Programming** Beim Pair Programming entwickeln zwei Personen die Software zusammen. Es gibt einen sogenannten Driver - die Person, die tippt - und einen sogenannten Navigator - die Person, die kritisch den Code überprüft. Das Pair Programming ist wichtiger in agilen Entwicklungsmethoden wie z. B dem Extreme Programming (XP).

**Refactoring** Unter Refactoring versteht man die Verbesserung der Code- und Systemstruktur mit dem Ziel einer besseren Wartbarkeit. Dabei sollen Lesbarkeit, Verständlichkeit, Wartbarkeit und Erweiterbarkeit verbessert werden, mit dem Ziel, den jeweiligen Aufwand für Fehleranalyse und funktionale Erweiterungen deutlich zu senken. Im Refactoring werden keine neuen Funktionalitäten programmiert.

**agile Softwareentwicklung** Agile Entwicklungsprozesse gehen auf das Agile Manifest, das im Jahr 2001 veröffentlicht wurde, zurück. Dieses Agile Manifest

bildet die Basis für agile Entwicklungsprozesse wie eXtreme Programming oder SCRUM. Das Manifest beinhaltet 12 Grundprinzipien für die moderne agile Software-Entwicklung. Hauptzielsetzung ist, rasch eine funktionsfähige Software-Lösung auszuliefern, um den Kundenwunsch bestmöglich zu erfüllen. Dabei spielt die Interaktion mit dem Kunden eine zentrale Rolle.

## 46116 / 2013 / Frühjahr / Thema 2 / Teilaufgabe 1 / Aufgabe 1

### Aufgabe 1

Gegeben sei folgendes Klassendiagramm:

- (a) Implementieren Sie die Klassen „Order“, „Payment“, „Check“, „OrderDetail“ und „Item“ in einer geeigneten objektorientierten Sprache Ihrer Wahl. Beachten Sie dabei insbesondere Sichtbarkeiten, Klassen- vs. Instanzzugehörigkeiten und Schnittstellen bzw. abstrakte Klassen.

Lösungsvorschlag

```
public class Check extends Payment {
    String bankName;
    long bankId;
    public boolean authorized() {
        return true;
    }
}
```

Code-Beispiel auf Github ansehen: [src/main/java/org/bschlangaul/examen/examen\\_46116/jahr\\_2013/fruehjahr/t2\\_ta1\\_a1/Check.java](https://github.com/bschlangaul/examen/examen_46116/jahr_2013/fruehjahr/t2_ta1_a1/Check.java)

```
@SuppressWarnings("unused")
public class Item {
    private double weight;
    public String description;

    public double getPrice() {
        return 42;
    }

    public double getWeight() {
        return 23;
    }
}
```

Code-Beispiel auf Github ansehen: [src/main/java/org/bschlangaul/examen/examen\\_46116/jahr\\_2013/fruehjahr/t2\\_ta1\\_a1/Item.java](https://github.com/bschlangaul/examen/examen_46116/jahr_2013/fruehjahr/t2_ta1_a1/Item.java)

```
import java.util.Date;

public class Order {
    public static double VAT = 0.19;

    protected Date date;
```

```
protected boolean shipped;

public double calcPrice() {
    return 0.1d;
}

public double calcWeight() {
    return 0.2d;
}

public double calcTax(double tax) {
    return 0.3d;
}
}
```

Code-Beispiel auf Github ansehen: [src/main/java/org/bschlangaul/examen/examen\\_46116/jahr\\_2013/fruehjahr/t2\\_ta1\\_a1/Order.java](#)

```
@SuppressWarnings("unused")
public class OrderDetail {
    private long quantity;

    Item item;

    public double calcPrice() {
        return 0.1d;
    }

    public double calcWeight() {
        return 0.2d;
    }
}
```

Code-Beispiel auf Github ansehen: [src/main/java/org/bschlangaul/examen/examen\\_46116/jahr\\_2013/fruehjahr/t2\\_ta1\\_a1/OrderDetail.java](#)

```
public abstract class Payment {
    double amount;
}
```

Code-Beispiel auf Github ansehen: [src/main/java/org/bschlangaul/examen/examen\\_46116/jahr\\_2013/fruehjahr/t2\\_ta1\\_a1/Payment.java](#)

(b) Erstellen Sie ein Sequenzdiagramm (mit konkreten Methodenaufrufen und Nummerierung der Nachrichten) für folgendes Szenario:

- (i) „Erika Mustermann“ aus „Rathausstraße 1, 10178 Berlin“ wird als neue Kundin angelegt.
- (ii) Frau Mustermann bestellt heute 1 kg Gurken und 2 kg Kartoffeln.
- (iii) Sie bezahlt mit ihrer Visa-Karte, die im August 2014 abläuft und die Nummer „1234 567891 23456“ hat — die Karte erweist sich bei der Prüfung als gültig.
- (iv) Am Schluss möchte sie noch wissen, wie viel ihre Bestellung kostet — dabei wird der Anteil der Mehrwertsteuer extra ausgewiesen.

## 46116 / 2014 / Frühjahr / Thema 2 / Teilaufgabe 1 / Aufgabe 1

Gegeben sei folgende Methode zur Berechnung der Anzahl der notwendigen Züge beim Spiel „Die Türme von Hanoi“:

```
int hanoi(int nr, char from, char to) {
    char free = (char) ('A' + 'B' + 'C' - from - to);
    if (nr > 0) {
        int moves = 1;
        moves += hanoi(nr - 1, from, free);
        System.out.println("Move piece nr. " + nr + " from " + from + " to " + to);
        moves += hanoi(nr - 1, free, to);
        return moves;
    } else {
        return 0;
    }
}
```

Code-Beispiel auf Github ansehen: [src/main/java/org/bschlangaul/examen/examen\\_46116/jahr\\_2014/fruehjahr/Hanoi.java](https://github.com/bschlangaul/examen/examen_46116/jahr_2014/fruehjahr/Hanoi.java)

- (a) Beweisen Sie formal mittels vollständiger Induktion, dass zum Umlegen von  $k$  Scheiben (z. B. vom Turm A zum Turm C) insgesamt  $2^k - 1$  Schritte notwendig sind, also dass für  $k \geq 0$  folgender Zusammenhang gilt:

$$\text{hanoi}(k, 'A', 'C') = 2^k - 1$$

Lösungsvorschlag

Zu zeigen:

$$\text{hanoi}(k, 'A', 'C') = 2^k - 1$$

### Induktionsanfang

— Beweise, dass  $A(1)$  eine wahre Aussage ist. \_\_\_\_\_

$$k = 0$$

$$\text{hanoi}(0, 'A', 'C') = 0$$

$$2^0 - 1 = 1 - 1 = 0$$

### Induktionsvoraussetzung

— Die Aussage  $A(k)$  ist wahr für ein beliebiges  $k \in \mathbb{N}$ . \_\_\_\_\_

$$\text{hanoi}(k, 'A', 'C') = 2^k - 1$$

**Induktionsschritt**

— Beweise, dass wenn  $A(n = k)$  wahr ist, auch  $A(n = k + 1)$  wahr sein muss.

$$\text{hanoi}(k, 'A', 'C') = 1 + \text{hanoi}(k - 1, 'A', 'B') + \text{hanoi}(k - 1, 'B', 'C')$$

$$k \rightarrow k + 1$$

$$\begin{aligned} \text{hanoi}(k + 1, 'A', 'C') &= 1 + \text{hanoi}((k + 1) - 1, 'A', 'B') + \\ &\quad \text{hanoi}((k + 1) - 1, 'B', 'C') \\ &= 1 + \text{hanoi}(k, 'A', 'B') + \\ &\quad \text{hanoi}(k, 'B', 'C') && k + 1 - 1 = k \\ &= 1 + 2^k - 1 + 2^k - 1 && \text{Formeln eingesetzt} \\ &= 2^k + 2^k - 1 && 1 - 1 - 1 = -1 \\ &= 2 \cdot 2^k - 1 && 2^k + 2^k = 2 \cdot 2^k \\ &= 2^{k+1} - 1 && 2 \cdot 2^k = 2^{k+1} \end{aligned}$$

- (b) Geben Sie eine geeignete Terminierungsfunktion an und begründen Sie kurz Ihre Wahl!

Lösungsvorschlag

Betrachte die Argumentenfolge  $k, k - 1, k - 2, \dots, 0$ . Die Terminierungsfunktion ist offenbar  $T(k) = k$ .  $T(k)$  ist bei jedem Rekursionsschritt auf der Folge der Argumente streng monoton fallend. Bei der impliziten Annahme  $k$  ist ganzzahlig und  $k \geq 0$  ist  $T(k)$  nach unten durch 0 beschränkt.

## 46116 / 2014 / Herbst / Thema 2 / Teilaufgabe 1 / Aufgabe 1

### Aufgabe 1: Allgemeine SWT, Vorgehensmodelle und Requirements Engineering

Kreuzen Sie für die folgenden Multiple-Choice-Fragen genau die richtigen Antworten deutlich an. Es kann mehr als eine Antwort richtig sein.

Jedes korrekt gesetzte oder korrekt nicht gesetzte Kreuz wird mit 1 Punkt gewertet. Jedes falsch gesetzte oder falsch nicht gesetzte Kreuz wird mit -1 Punkt gewertet. Eine Frage kann entwertet werden, dann wird sie nicht in der Korrektur berücksichtigt. Einzelne Antworten können nicht entwertet werden. Entwerten Sie eine Frage wie folgt

Die gesamte Aufgabe wird nicht mit weniger als 0 Punkten gewertet.

(a) Welche Aussage ist wahr?

- ☐ Je früher ein Fehler entdeckt wird, umso teurer ist seine Korrektur.
- ☐ Je später ein Fehler entdeckt wird, umso teurer ist seine Korrektur.
- ☐ Der Zeitpunkt der Entdeckung hat keinen Einfluss auf die Kosten.

Lösungsvorschlag

2 ist richtig: Je später der Fehler entdeckt wird, desto mehr wurde er schon in das Projekt „eingearbeitet“, daher dauert das Beseitigen des Fehlers länger und das kostet mehr Geld.

(b) Mit welcher Methodik können Funktionen spezifiziert werden?

- ☐ Als Funktionsvereinbarung in einer Programmiersprache
- ☐ Mit den Vor- und Nachbedingungen von Kontrakten
- ☐ Als Zustandsautomaten

Lösungsvorschlag

2 und 3 ist richtig: Die Spezifikation soll unabhängig von einer Programmiersprache sein.

(c) Welche Vorgehensmodelle sind für Projekte mit häufigen Änderungen geeignet?

- ☐ Extreme Programming (XP)
- ☐ Das V-Modell 97
- ☐ Scrum

Lösungsvorschlag

1 und 3 ist richtig. Das V-Modell ist ein starres Vorgehensmodell, bei dem alle Anforderungen zu Beginn vorhanden sein müssen.

(d) Welche der folgenden Aussagen ist korrekt?

- ☐ Mittels Prototyping versucht man die Anzahl an nötigen Unit-Tests zu reduzieren.
- ☐ Ein Ziel von Prototyping ist die Erhöhung der Qualität während der Anforderungsanalyse.
- ☐ Mit Prototyping versucht man sehr früh Feedback von Stakeholdern zu erhalten.

2 und 3 ist richtig: Prototypen müssen auch getestet werden. Es kann nicht an Tests gespart werden. Durch das häufige Feedback des Kunden / der Stakeholder können die Anforderungen immer genauer und klarer erfasst werden.

(e) Welche der folgenden Aussagen ist korrekt?

- ☐ Bei der Architektur sollten funktionale und nicht-funktionale Anforderungen beachtet werden.
- ☐ Bei der Architektur sollten nur funktionale Anforderungen beachtet werden.
- ☐ Bei der Architektur sollten nur nicht-funktionale Anforderungen beachtet werden.
- ☐ Bei der Architektur sollte auf die mögliche Änderungen von Komponenten geachtet werden.

1 und 4 ist richtig: Mögliche Änderungen werden durch klar definierte Schnittstellen und wenig Kopplung der Komponenten erleichtert. (Kopplung handelt von Abhängigkeiten zwischen Modulen. Kohäsion handelt von Abhängigkeiten zwischen Funktionen innerhalb eines Moduls.)

## 46116 / 2014 / Herbst / Thema 2 / Teilaufgabe 1 / Aufgabe 2

Software wird oft in definierten Prozessen entwickelt. Diese nennt man Vorgehensmodelle.

Allgemein

(a) Was sind die Aufgaben eines Vorgehensmodells im Allgemeinen?

- liefert Erfahrungen und bewährte Methoden
- beschreibt die am Projekt beteiligten Rollen
- legt Aufgaben und Aktivitäten fest
- definiert einheitliche Begriffe
- gibt Techniken, Werkzeuge, Richtlinien / Standards an

(b) Was sind die wesentlichen Bestandteile eines Vorgehensmodells und in welcher Beziehung stehen diese zueinander?

Bestandteile: Anforderungsanalyse, Modellierung, Implementierung, Test, Auslieferung, Wartung

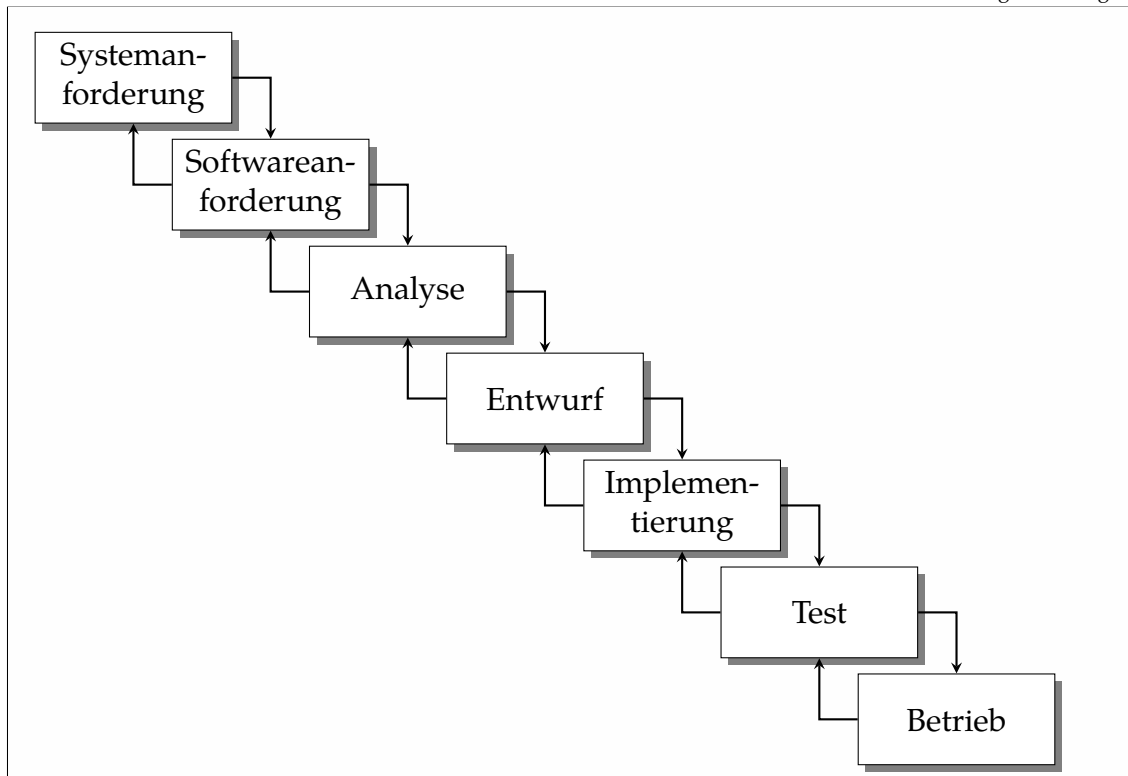
Die einzelnen Phasen bauen immer aufeinander aus. Je nach Vorgehensmodell können sich Phasen auch wiederholen (Prototyping, Scrum...).

Wasserfallmodell

Ein frühes Vorgehensmodell ist das von Dr. Winston Royce 1970 formalisierte Wasserfallmodell.

(a) Geben Sie eine schematische Darstellung des Wasserfallmodells an.

Lösungsvorschlag



(b) Nennen Sie zwei Probleme des Modells und erläutern Sie diese kurz.

Lösungsvorschlag

Fehler werden ggf. erst am Ende des Entwicklungsprozesses erkannt, da erst dort das Testen stattfindet. Dadurch kann die Behebung eines Fehlers sehr aufwändig und somit teuer werden.

Der Kunde / Endanwender wird erst nach der Implementierung wieder eingebunden. Das bedeutet, dass er nach der Stellung der Anforderungen keinen Einblick mehr in den Prozess hat und somit auch nicht gegensteuern kann, falls ihm etwas nicht gefällt oder er etwas nicht bedacht hat.

(c) In welchen Situationen lässt sich das Wasserfallmodell gut einsetzen?

Lösungsvorschlag

Das Wasserfallmodell ist geeignet, wenn es sich um ein von Anfang an klar definiertes Projekt ohne große Komplexität handelt, bei dem alle Anforderungen, Aufwand und Kosten schon zu Beginn des Projekts feststehen bzw.



abgeschätzt werden können.

Barry Boehm erweiterte das Wasserfallmodell 1979 zum so genannten V-Modell.

- (a) Geben Sie eine schematische Darstellung des V-Modells an.
- (b) Nennen Sie zwei Probleme des Modells und erläutern Sie diese kurz.

Lösungsvorschlag

- Die Nachteile des Wasserfallmodells bestehen weiterhin!
- Nicht für kleine Projekte geeignet, da aufwändige Tests vorgesehen sind, die im Kleinen detailliert meist nicht stattfinden (können).

- (c) Welchen Vorteil hat das V-Modell gegenüber dem Wasserfallmodell?

Lösungsvorschlag

Für jedes Dokument besteht ein entsprechender Test (Validierung / Verifikation). Dabei kann die Planung der Tests schon vor der eigentlichen Durchführung geschehen, so dass Aktivitäten im Projektteam parallelisiert werden können. So kann zum Beispiel der Tester die Testfälle für den Akzeptanztest (=Test des Systementwurfs) entwickeln, auch wenn noch keine Implementierung existiert.

In neuerer Zeit finden immer häufiger iterative und inkrementelle Vorgehensweisen Anwendung.

- (a) Erklären Sie den Begriff iterative Softwareentwicklung.

Lösungsvorschlag

Iterativ heißt, dass der Entwicklungsprozess mehrfach wiederholt wird: statt den „Wasserfall“ einmal zu durchlaufen, werden „kleine Wasserfälle“ hintereinander gesetzt.

- (b) Erklären Sie den Begriff inkrementelle Softwareentwicklung und grenzen Sie ihn von iterativer Softwareentwicklung ab.

Lösungsvorschlag

Bei der inkrementellen Entwicklung wird das System Schritt für Schritt fertig gestellt. D. h., dass ein Prototyp immer etwas mehr kann als der Prototyp davor. Dies wird durch die iterative Entwicklung unterstützt, da bei jeder Wiederholung des Entwicklungsprozesses ein neues Inkrement entsteht, d. h. ein neuer Prototyp, der mehr Funktionalitäten benutzt als der vorangegangene.

- (c) Nennen Sie jeweils zwei Vor- und Nachteile eines iterativen und inkrementellen Vorgehens im Vergleich zum Wasserfallmodell.

**Vorteile:**

- Risiken können früher erkannt werden.
- volatile Anforderungen können besser berücksichtigt werden.
- inkrementelle Auslieferung wird erleichtert.

**Nachteile:**

- komplexeres Projektmanagement
- schwerer messbar
- (Mehrarbeit)<sup>a</sup>

<sup>a</sup>Quelle: <https://www.pst.ifi.lmu.de/Lehre/WS0607/pm/vorlesung/PM-02-Prozess.pdf>

**46116 / 2014 / Herbst / Thema 2 / Teilaufgabe 1 / Aufgabe 3**

Gegeben sei folgender Sachverhalt:

Für eine Verwaltungssoftware einer Behörde soll ein Bestellsystem entwickelt werden. Dabei sollen die Nutzer ihre Raummaße eingeben können. Anschließend können die Nutzer über ein Web-Interface das Büro gestalten und Möbel (wie zum Beispiel Wandschränke) und andere Einrichtungsgegenstände in einem virtuellen Büro platzieren. Aus dem Web-Interface kann die Einrichtung dann direkt bestellt werden. Dazu müssen die Nutzer ihre Büro-Nummer und den Namen und die Adresse der Behörde eingeben und die Bestellung bestätigen.

Weiterhin können Nutzer auch Büromaterialien über das Web-Interface bestellen. Dazu ist anstatt der Eingabe der Raummaße nur das Eingeben von Büro-Nummer und des Namens und der Adresse der Behörde erforderlich.

Zusätzlich zum Standard-Nutzer können sich auch Administratoren im System anmelden und Möbel zur Kollektion hinzufügen und aus der Kollektion entfernen. Die Möbel können eindeutig durch ihre Inventurnummer identifiziert werden.

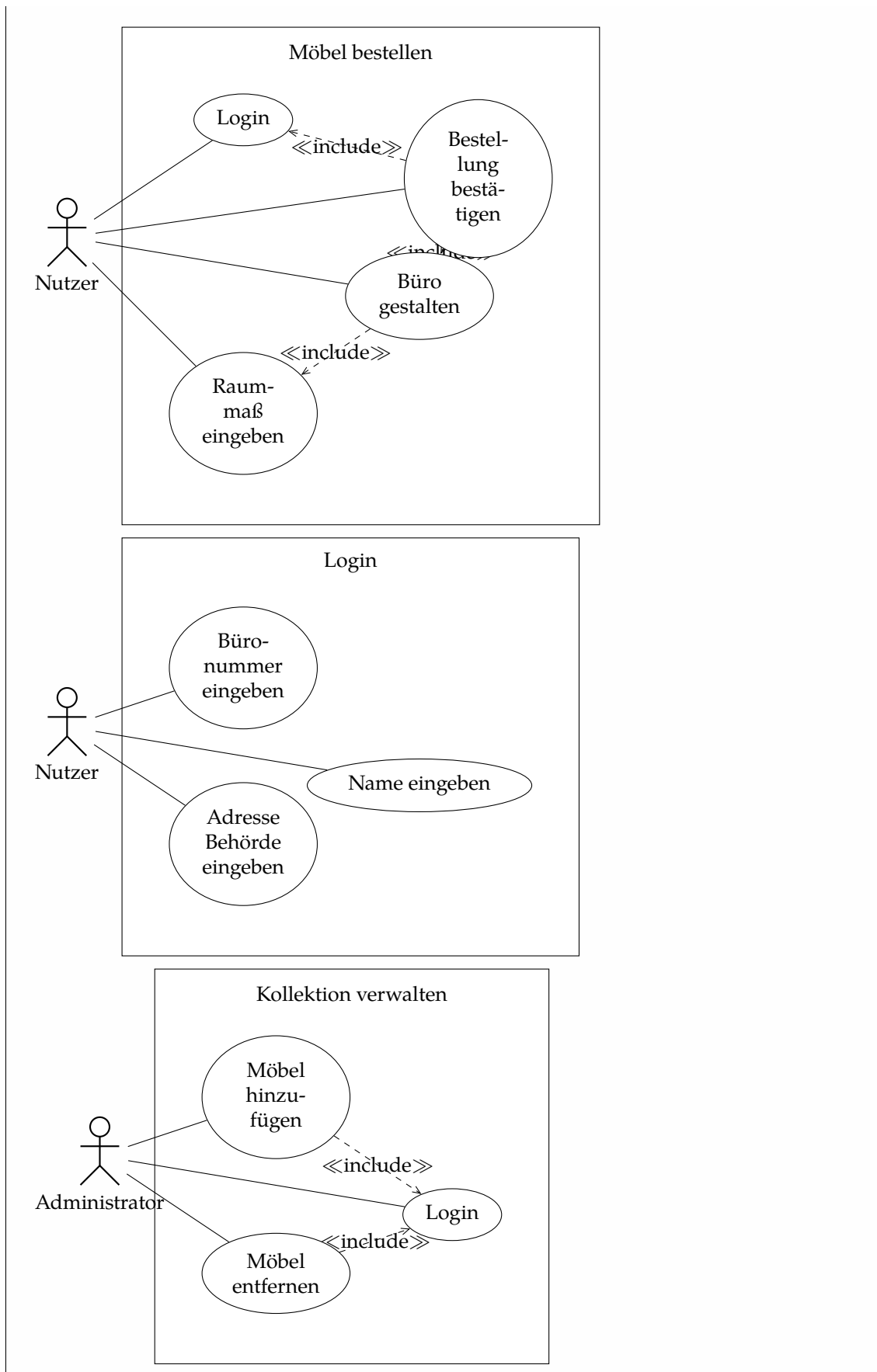
Um jegliche Veränderungen im System protokollieren zu können müssen Nutzer und Administratoren zur Bestätigung eingeloggt sein.

- (a) Erfassen Sie die drei Systemfunktionen *Möbel bestellen*, *Login* und *Kollektion verwalten* in einem UML-konformen Use Case Diagramm.

Login

Kollektion verwalten

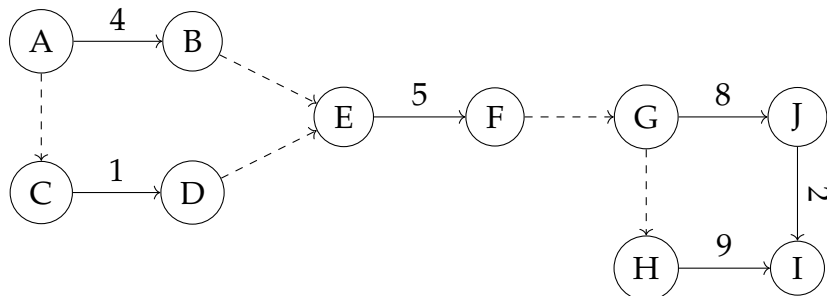
Lösungsvorschlag



- (b) Erstellen Sie ein UML-Klassendiagramm, welches die Beziehungen und sinnvolle Attribute der Klassen „Nutzer, Büro, Möbelstück und Wandschrank“ darstellt.
- (c) Instanzieren Sie das Klassendiagramm in einem Objektdiagramm mit den zwei Nutzern mit Namen Ernie und Bernd, einem Büro mit der Nummer CAPITOL2 und zwei Schränken mit den Inventurnummern S1.88 und S1.77.
- (d) Geben Sie für ein Büromöbelstück ein Zustandsdiagramm an. Überlegen Sie dazu, welche möglichen Zustände ein Möbelstück während des Bestellvorgangs haben kann und finden Sie geeignete Zustandsübergänge.

### 46116 / 2015 / Frühjahr / Thema 1 / Teilaufgabe 1 / Aufgabe 3

Betrachten Sie folgendes CPM-Netzwerk:



- (a) Berechnen Sie die früheste Zeit für jedes Ereignis, wobei angenommen wird, dass das Projekt zum Zeitpunkt 0 startet.

Lösungsvorschlag

$i$	Nebenrechnung	$FZ_i$
A		0
B		4
C		0
D		1
E	$\max(4, 1)$	4
F		9
G		9
H		9
J		17
I	$\max(9_{(\rightarrow H)} + 9, 17_{(\rightarrow J)} + 2)$	19

- (b) Setzen Sie anschließend beim letzten Ereignis die späteste Zeit gleich der frühesten Zeit und berechnen Sie die spätesten Zeiten.

Lösungsvorschlag

Gantt-Diagramm  
Gantt-Diagramm

$i$	Nebenrechnung	$SZ_i$
A	$\min(3,0)$	0
B		4
C		3
D		4
E		4
F		9
G	$\min(10,9)$	9
H		10
J		17
I		19

(c) Berechnen Sie nun für jedes Ereignis die Pufferzeiten.

Lösungsvorschlag

$i$	A	B	C	D	E	F	G	H	J	I
$FZ_i$	0	4	0	1	4	9	9	9	17	19
$SZ_i$	0	4	3	4	4	9	9	10	17	19
GP	0	0	3	3	0	0	0	1	0	0

(d) Bestimmen Sie den kritischen Pfad.

Lösungsvorschlag

$A \rightarrow B \rightarrow E \rightarrow F \rightarrow G \rightarrow J \rightarrow I$
---

(e) Was ist ein Gantt-Diagramm? Worin unterscheidet es sich vom CPM-Netzwerk?

Lösungsvorschlag

--

**46116 / 2015 / Herbst / Thema 1 / Teilaufgabe 1 / Aufgabe 3**

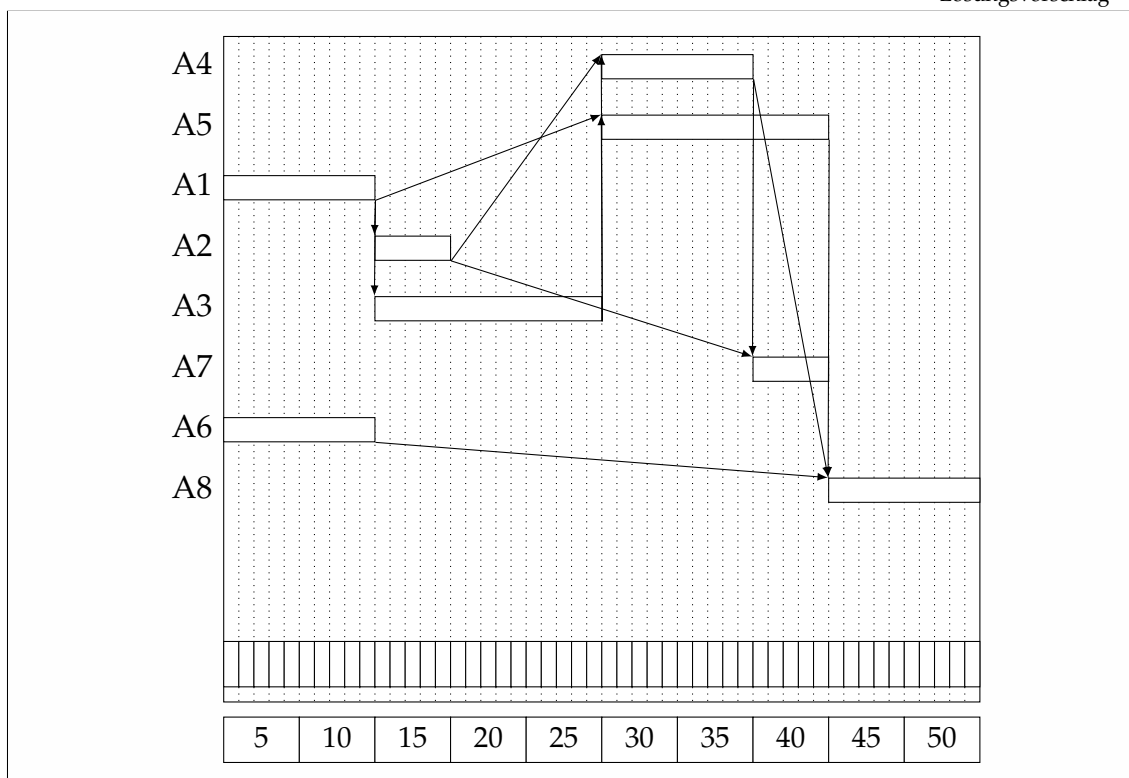
Betrachten Sie die folgende Tabelle zum Projektmanagement:

Name	Dauer (Tage)	Abhängig von
A1	10	
A2	5	A1
A3	15	A1
A4	10	A2, A3
A5	15	A1, A3
A6	10	
A7	5	A2, A4
A8	10	A4, A5, A6

Tabelle 1: Übersicht Arbeitspakete

- (a) Erstellen Sie ein Gantt-Diagramm, das die in der Tabelle angegebenen Abhängigkeiten berücksichtigt.

Lösungsvorschlag



- (b) Wie lange dauert das Projekt mindestens?
- (c) Geben Sie den oder die kritischen Pfad(e) an.
- (d) Konstruieren Sie ein PERT-Chart zum obigen Problem.

## 46116 / 2015 / Herbst / Thema 2 / Teilaufgabe 1 / Aufgabe 2

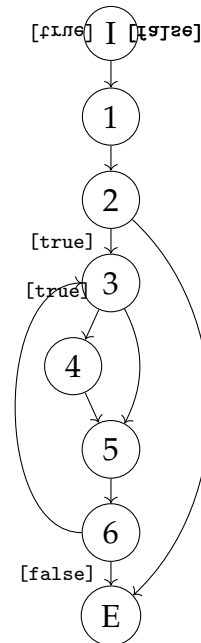
Gegeben sei folgende Methode `isPalindrom` und ihr Kontrollflussgraph:

Abkürzungen: I = Import, E = Export

```
boolean isPalindrom(String s) {
    boolean yesItIs = true;
    if (s != null && s.length() > 1) {
        do {
            if (s.charAt(0) != s.charAt(s.length() -
↪ 1)) {
                yesItIs = false;
            }
            s = s.substring(1, s.length() - 1);

        } while (yesItIs && s.length() > 1);
    }
    return yesItIs;
}
```

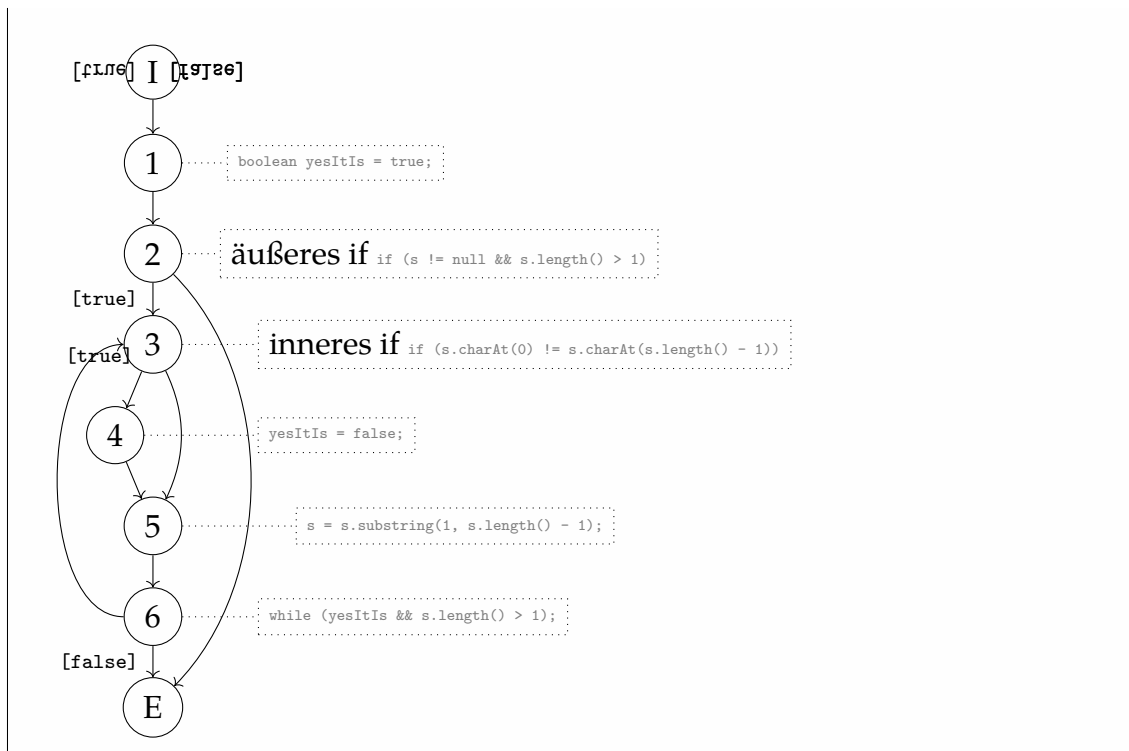
Code-Beispiel auf Github ansehen:  
[src/main/java/org/bschlangaul/aufgaben/sosy/pu\\_5/Aufgabe2.java](https://github.com/bschlangaul/aufgaben/sosy/pu_5/Aufgabe2.java)



- (a) Geben Sie je einen Repräsentanten aller Pfadklassen **im Kontrollflussgraphen** an, die zum Erzielen einer vollständigen ... mit **minimaler** Testfallanzahl und **möglichst kurzen** Pfaden genügen würden.

Lösungsvorschlag

Bemerkung: In der Aufgabenstellung steht „Geben Sie je einen Repräsentanten aller Pfadklassen **im Kontrollflussgraphen** an, [...]“. Das bedeutet, dass es hier erstmal egal ist, ob ein Pfad im Code möglich ist oder nicht!



C1-Test Zweigüberdeckung  
(Branch Coverage)  
C2b Schleife-Inneres-  
Pfadüberdeckung  
(Boundary-Interior Path  
Coverage)

(i) Verzweigungsüberdeckung (Branch-Coverage,  $C_1$ )

Lösungsvorschlag

**Pfad 1 (p1)** ① - ① - ② - ⑤

(äußere **if**-Bedingung **false**)

**Pfad 2 (p2)** ① - ① - ② - ③ - ⑤ - ⑥ - ③ - ④ - ⑤ - ⑥ - ⑤

(äußere **if**-Bedingung **true**, innere **if**-Bedingung **false**, Wiederholung, innere **if**-Bedingung **true**, keine Wiederholung)

(ii) Schleife-Inneres-Überdeckung (Boundary-Interior-Coverage,  $C_{\infty,2}$ )



**ohne Ausführung der Wiederholung (äußere Pfade):** p1 (siehe oben)

① - ① - ② - ⑥

**Boundary-Test:** (alle Pfade, die die Wiederholung betreten, aber nicht wiederholen; innerhalb des Schleifenrumpfes alle Pfade!)

**interior-Test:** (alle Pfade mit *einer Wiederholung des Schleifenrumpfes*; innerhalb des Schleifenrumpfes wieder alle Pfade!)

innere **if**-Bedingung **true**: ③ - ④ - ⑤ - ⑥

innere **if**-Bedingung **false**: ③ - ⑤ - ⑥

**p5** ① - ① - ② - ③ - ④ - ⑤ - ⑥ - ③ - ④ - ⑤ - ⑥ - ⑥

(innere **if**-Bedingung **true**, innere **if**-Bedingung **true**)

**p2** (siehe oben) ① - ① - ② - ③ - ⑤ - ⑥ - ③ - ④ - ⑤ - ⑥ - ⑥

(innere **if**-Bedingung **false**, innere **if**-Bedingung **true**)

**p6** ① - ① - ② - ③ - ④ - ⑤ - ⑥ - ③ - ④ - ⑤ - ⑥ - ⑥

(innere **if**-Bedingung **true**, innere **if**-Bedingung **false**)

p7 ① - ① - ② - ③ - ⑤ - ⑥ - ③ - ⑤ - ⑥ - ⑥  
 (innere **if**-Bedingung **false**, innere **if**-Bedingung **false**)

Zyklomatische Komplexität  
 nach Mc-Cabe  
 C2a Vollständige  
 Pfadüberdeckung (Full  
 Path Coverage)

mit **minimaler** Testfallanzahl und **möglichst kurzen** Pfaden genügen würden.

- (b) Welche der vorangehend ermittelten Pfade für die  $C_{\infty,2}$ -Überdeckung sind mittels Testfällen tatsächlich überdeckbar („feasible“)? Falls der Pfad ausführbar ist, geben Sie den zugehörigen Testfall an - andernfalls begründen Sie kurz, weshalb der Pfad nicht überdeckbar ist.

Lösungsvorschlag

p1 s = "a";  
 p2 s = "abaa";  
 p3 s = "ab";  
 p4 s = "aa";  
 p5 nicht überdeckbar, da **yesItIs** = **false**, wenn innere **if**-Bedingung **t**  
**rue**) keine Wiederholung!  
 p6 nicht überdeckbar, da **yesItIs** = **false**, wenn innere **if**-Bedingung **t**  
**rue**) keine Wiederholung!  
 p7 s = "abba";

- (c) Bestimmen Sie anhand des Kontrollflussgraphen des obigen Code-Fragments die maximale Anzahl linear unabhängiger Programmpfade, also die zyklomatische Komplexität nach McCabe.

Lösungsvorschlag

$M = b + p = 3 + 1 = 4$   
 (b: Anzahl Binärverzweigungen, p: Anzahl Zusammenhangskomponenten)  
**Alternativ**  
 $M = e - n + 2p = 10 - 8 + 2 = 4$   
 (e: Anzahl Kanten, n: Anzahl Knoten, p: Anzahl Zusammenhangskomponenten)

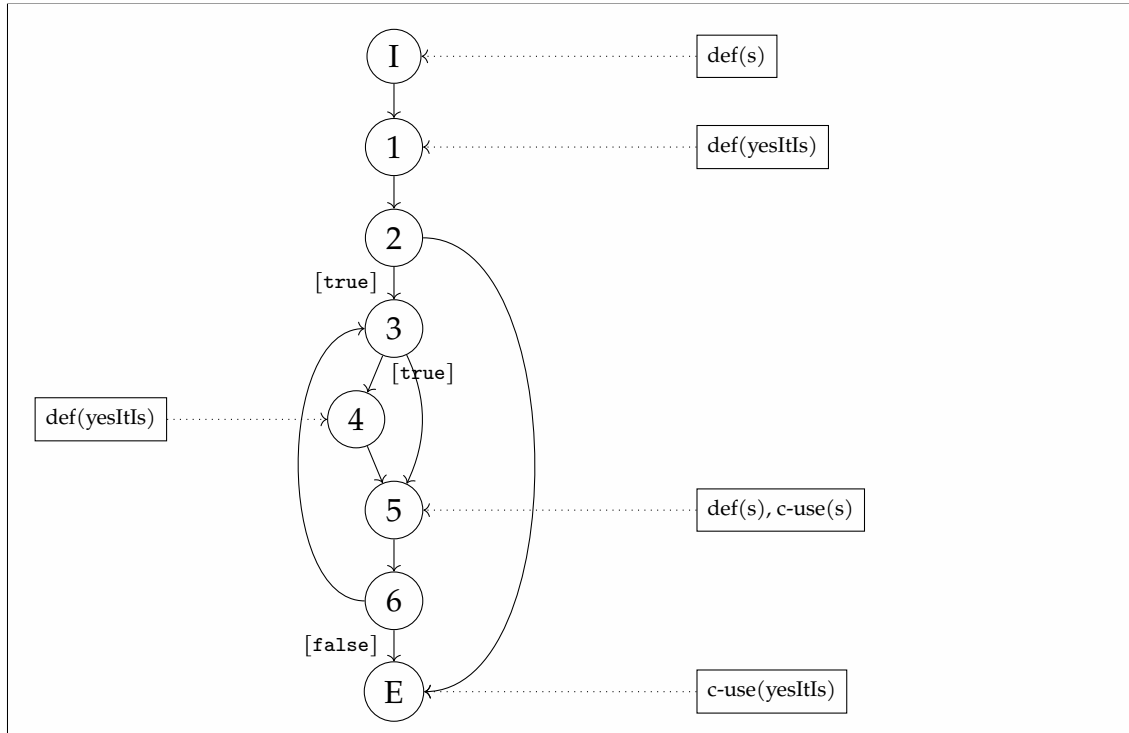
- (d) Kann für dieses Modul eine 100%-ige Pfadüberdeckung erzielt werden? Begründen Sie kurz Ihre Antwort.

Lösungsvorschlag

Eine 100%-ige Pfadüberdeckung kann nicht erzielt werden, da es zum einen unüberdeckbare Pfade gibt (vgl. Teilaufgabe b). Zum anderen ist das Testen aller Testfälle nicht möglich, da die Anzahl an Zeichen des übergebenen Wortes nicht begrenzt ist und es somit eine unendliche Anzahl an Testfällen gibt.

- (e) Übernehmen Sie den vorgegebenen Kontrollflussgraphen und annotieren Sie ihn mit allen relevanten Datenflussereignissen. Geben Sie jeweils an, ob die Verwendungen berechnend (c-use) oder prädikativ (p-use) sind.

Lösungsvorschlag



### 46116 / 2015 / Herbst / Thema 2 / Teilaufgabe 1 / Aufgabe 3

Sei  $wp(A, Q)$  die schwächste Vorbedingung (weakest precondition) eines Programmfragments  $A$  bei gegebener Nachbedingung  $Q$  so, dass  $A$  alle Eingaben, die  $wp(A, Q)$  erfüllen, auf gültige Ausgaben abbildet, die  $Q$  erfüllen.

Bestimmen Sie schrittweise und formal (mittels Floyd-Hoare-Kalkül) jeweils  $wp(A, Q)$  für folgende Code-Fragmente  $A$  und Nachbedingungen  $Q$  und vereinfachen Sie dabei den jeweils ermittelten Ausdruck so weit wie möglich.

Die Variablen  $x, y$  und  $z$  in folgenden Pseudo-Codes seien ganzzahlig (vom Typ `int`). Zur Vereinfachung nehmen Sie bitte im Folgenden an, dass die verwendeten Datentypen unbeschränkt sind und daher keine Überläufe auftreten können.

- (a) Sequenz:

```

x = -2 * (x + 2 * y);
y += 2 * x + y + z;
z -= x - y - z;

```

$Q \equiv x = y + z$

Lösungsvorschlag

Code umformulieren:

```

x = -2 * (x + 2 * y);

```

$$\begin{aligned} y &= y + 2 * x + y + z; \\ z &= z - (x - y - z); \end{aligned}$$

$$\text{wp}("x=-2*(x+2*y); y=2*y+2*x+z; z=z-(x-y-z);", x = y + z)$$

z einsetzen

$$\equiv \text{wp}("x=-2*(x+2*y); y=2*y+2*x+z;", x = y + (z - (x - y - z)))$$

Innere Klammer auflösen

$$\equiv \text{wp}("x=-2*(x+2*y); y=2*y+2*x+z;", x = y + (-x + y - 2z))$$

Klammer auflösen

$$\equiv \text{wp}("x=-2*(x+2*y); y=2*y+2*x+z;", x = -x + 2y + 2z)$$

-x auf beiden Seiten

$$\equiv \text{wp}("x=-2*(x+2*y); y=2*y+2*x+z;", 0 = -2x + 2y + 2z)$$

÷2 auf beiden Seiten

$$\equiv \text{wp}("x=-2*(x+2*y); y=2*y+2*x+z;", 0 = -x + y + z)$$

y einsetzen

$$\equiv \text{wp}("x=-2*(x+2*y);", 0 = -x + (2y + 2x + z) + z)$$

Term vereinfachen

$$\equiv \text{wp}("x=-2*(x+2*y);", 0 = x + 2y + 2z)$$

x einsetzen

$$\equiv \text{wp}("", 0 = (-2(x + 2y)) + 2y + 2z)$$

wp weglassen

$$\equiv 0 = (-2(x + 2y)) + 2y + 2z$$

ausmultiplizieren

$$\equiv 0 = (-2x - 4y) + 2y + 2z$$

Klammer auflösen, vereinfachen

$$\equiv 0 = -2x - 2y + 2z$$

÷2 auf beiden Seiten

$$\equiv 0 = -x - y + z$$

$x$  nach links holen mit  $+x$  auf beiden Seiten

$$\equiv x = -y + z$$

$y$  ganz nach links schreiben

$$\equiv x = z - y$$

$$x = -2 \cdot (x + 2 \cdot y)$$

(b) Verzweigung:

```
if (x < y) {
  x = y + z;
} else if (y > 0) {
  z = y - 1;
} else {
  x -= y -= z;
}
```

$$Q \equiv x > z$$

Lösungsvorschlag

**1. Fall:**  $x < y$

**2. Fall:**  $x \geq y \wedge y > 0$

**3. Fall:**  $x \geq y \wedge y \leq 0$

Code umformulieren:

```
if (x < y) {
  x = y + z;
} else if (x >= y && y > 0) {
  z = y - 1;
} else {
  y = y - z;
  x = x - y;
}
```

$$\text{wp}(\text{"if}(x < y)\{x=y+z;\}\text{else if}(x \geq y \&\& y > 0)\{z=y-1;\}\text{else}\{y=y-z;x=x-y;\}\text{"}, x > z)$$

$\equiv$

(In mehrere kleinere wp-Kalküle aufsplitten)

$$\begin{aligned} & \left( (x < y) \wedge \text{wp}("x=y+z;", x > z) \right) \vee \\ & \left( (x \geq y \wedge y > 0) \wedge \text{wp}("z=y-1;", x > z) \right) \vee \\ & \left( (x \geq y \wedge y \leq 0) \wedge \text{wp}("y=y-z;x=x-y;", x > z) \right) \end{aligned}$$

$\equiv$

(Code einsetzen)

$$\begin{aligned}
& \left( (x < y) \wedge \text{wp}("", y + z > z) \right) \vee \\
& \left( (x \geq y \wedge y > 0) \wedge \text{wp}("", x > y - 1) \right) \vee \\
& \left( (x \geq y \wedge y \leq 0) \wedge \text{wp}("y=y-z;", x - y > z) \right)
\end{aligned}$$

≡

(wp-Kalkül-Schreibweise weg lassen, Code weiter einsetzen)

$$\begin{aligned}
& \left( (x < y) \wedge y + z > z \right) \vee \\
& \left( (x \geq y \wedge y > 0) \wedge x > y - 1 \right) \vee \\
& \left( (x \geq y \wedge y \leq 0) \wedge \text{wp}("", x - (y - z) > z) \right)
\end{aligned}$$

≡

(Terme vereinfachen, wp-Kalkül-Schreibweise weg lassen)

$$\begin{aligned}
& \left( x < y \wedge y > 0 \right) \vee \\
& \left( x \geq y \wedge y > 0 \right) \vee \\
& \left( (x \geq y \wedge y \leq 0) \wedge x - (y - z) > z \right)
\end{aligned}$$

≡

(letzten Term vereinfachen)

$$\begin{aligned}
& \left( x < y \wedge y > 0 \right) \vee \\
& \left( x \geq y \wedge y > 0 \right) \vee \\
& \left( (x \geq y \wedge y \leq 0) \wedge x - y > 0 \right)
\end{aligned}$$

≡

(ein  $\wedge$  eliminieren)

$$\begin{aligned} & (x < y \wedge y > 0) \vee \\ & (x \geq y \wedge y > 0) \vee \\ & (y \leq 0 \wedge x > y) \end{aligned}$$

---

$^a x > y - 1 \wedge x \geq y$  ergibt  $x \geq y$   
 $^a x > y - 1 \wedge x \geq y$  ergibt  $x \geq y$

(c) Mehrfachauswahl:

```
switch (z) {
  case "x":
    y = "x";
  case "y":
    y = --z;
    break;
  default:
    y = 0x39 + "?";
}
```

$Q \equiv 'x' = y$

Hinweis zu den ASCII-Codes

- 'x' = 120<sub>(10)</sub>
- 'y' = 121<sub>(10)</sub>
- 0x39 = 57<sub>(10)</sub>
- '?' = 63<sub>(10)</sub>

Lösungsvorschlag

Mehrfachauswahl in Bedingte Anweisungen umschreiben. Dabei beachten, dass bei fehlendem **break** die Anweisungen im folgenden Fall bzw. ggf. in den folgenden Fällen ausgeführt werden:

```
if (z == "x") {
  y = "x";
  y = z - 1;
} else if (z == "y") {
  y = z - 1;
} else {
  y = 0x39 + "?";
}
```

Da kein **break** im Fall  $z == "x"$ .  $--z$  bedeutet, dass die Variable erst um eins verringert und dann zugewiesen wird.

```

if (z == 120) {
  y = 120;
  y = 120 - 1;
} else if (z == 121) {
  y = 121 - 1;
} else {
  y = 57 + 63;
}

```

Vereinfachung / Zusammenfassung:

```

if (z == 120) {
  y = 120;
  y = 119;
} else if (z == 121) {
  y = 120;
} else {
  y = 120;
}

```

$\text{wp}(\text{"if}(z==120)\{y=120;y=119;\}\text{else if}(z==121)\{y=120;\}\text{else}\{y=120;\}\text{"}, 120 = y)$

$\equiv$

(In mehrere kleinere wp-Kalküle aufsplitten)

$$\begin{aligned}
 & \left( (z = 120) \wedge \text{wp}(\text{"y=120;y=119;"}, 120 = y) \right) \vee \\
 & \left( ((z \neq 120) \wedge (z = 121)) \wedge \text{wp}(\text{"y=120;"}, 120 = y) \right) \vee \\
 & \left( ((z \neq 120) \wedge (z \neq 121)) \wedge \text{wp}(\text{"y=120;"}, 120 = y) \right)
 \end{aligned}$$

$\equiv$

(Code einsetzen)

$$\begin{aligned}
 & \left( (z = 120) \wedge \text{wp}(\text{"y=120;"}, 120 = 119) \right) \vee \\
 & \left( ((z \neq 120) \wedge (z = 121)) \wedge \text{wp}(\text{"", 120 = 120}) \right) \vee \\
 & \left( ((z \neq 120) \wedge (z \neq 121)) \wedge \text{wp}(\text{"", 120 = 120}) \right)
 \end{aligned}$$

$\equiv$

(vereinfachen)



$$\begin{aligned}
 & \text{false} \vee \\
 & \left( (z = 121) \wedge \text{true} \right) \vee \\
 & \left( ((z \neq 120) \wedge (z \neq 121)) \wedge \text{true} \right)
 \end{aligned}$$

$$\equiv \text{false} \vee (z = 121) \vee ((z \neq 120) \wedge (z \neq 121))$$

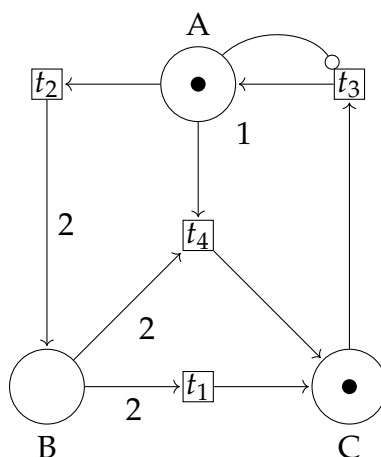
$$\equiv (z = 121) \vee (z \neq 121)$$

$$\equiv z \neq 121$$

Alle Zahlen außer 120 sind möglich bzw. alle Zeichen außer 'x'.

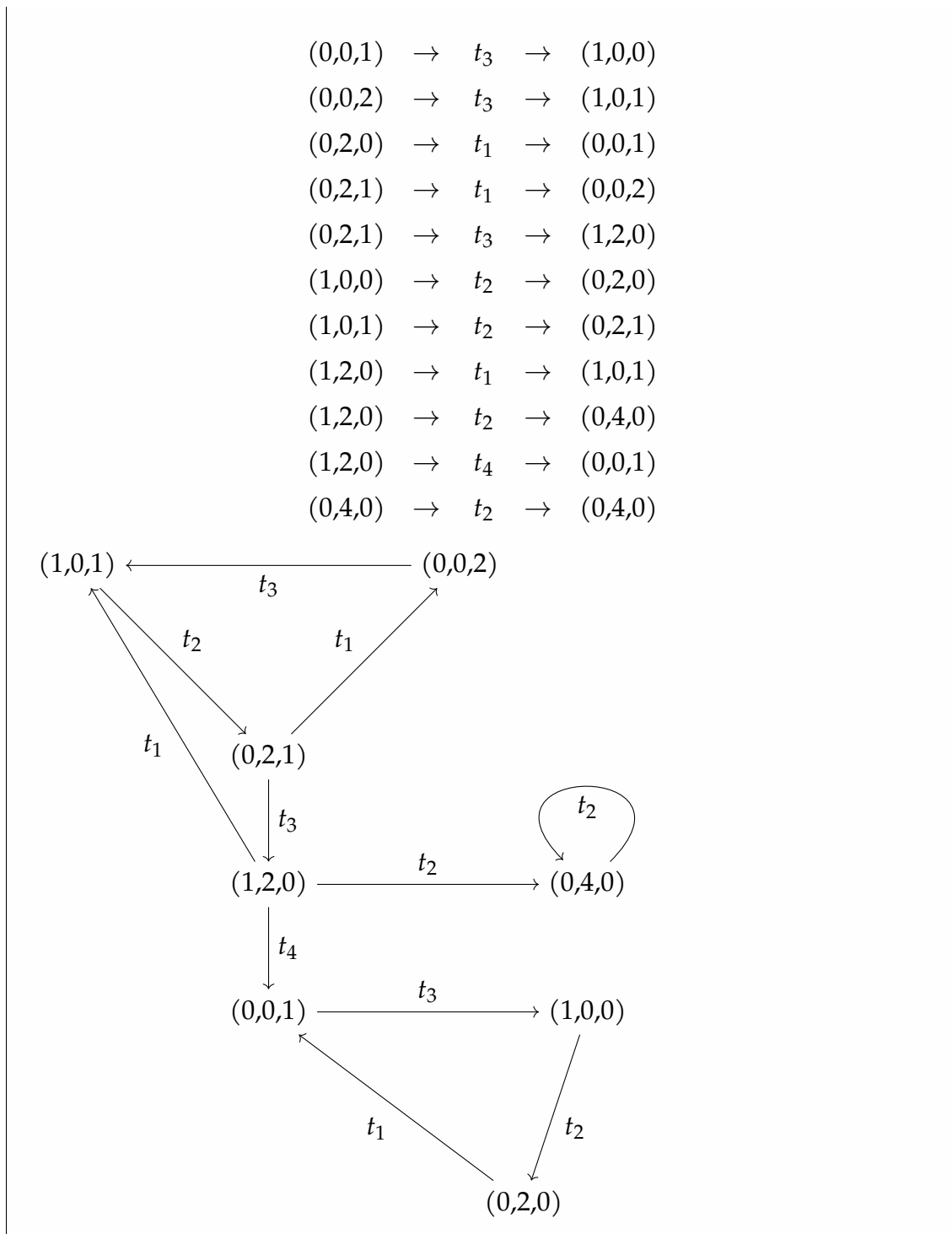
## 46116 / 2016 / Frühjahr / Thema 2 / Teilaufgabe 1 / Aufgabe 2

Gegeben sei das folgende Petri-Netz:



- (a) Erstellen Sie den zum Petri-Netz gehörenden Erreichbarkeitsgraphen. Die Belegungen sind jeweils in der Form  $[A, B, C]$  anzugeben. Beschriften Sie auch jede Kante mit der zugehörigen Transition. Beachten Sie die auf 1 beschränkte Kapazität von Stelle A oder alternativ die Inhibitor-Kante von A zu  $t_3$  (beides ist hier semantisch äquivalent).

Lösungsvorschlag



- (b) Wie kann man mit Hilfe des Erreichbarkeitsgraphen feststellen, ob ein Petri-Netz lebendig ist?
- (c) Aufgrund von Transition  $t_4$  ist das gegebene Petri-Netz nicht stark lebendig. Wie müssten die Pfeilgewichte der Transition  $t_4$  verändert werden, damit das Petri-Netz mit der gegebenen Startmarkierung beschränkt bleibt und lebendig wird?

$t_4$  nach C mit Gewicht 2 versehen

## 46116 / 2016 / Herbst / Thema 2 / Teilaufgabe 1 / Aufgabe 4

Gegeben sei folgende rekursive Methodendeklaration in der Sprache Java. Es wird als Vorbedingung vorausgesetzt, dass die Methode `cn` nur für Werte  $n \geq 0$  aufgerufen wird.

```
int cn(int n) {
    if (n == 0)
        return 1;
    else
        return (4 * (n - 1) + 2) * cn(n - 1) / (n + 1);
}
```

Sie können im Folgenden vereinfachend annehmen, dass es keinen Überlauf in der Berechnung gibt, sodass der Datentyp `int` für die Berechnung des Ergebnisses stets ausreicht.

- (a) Beweisen Sie mittels vollständiger Induktion, dass der Methodenaufruf `cn(n)` für jedes  $n \geq 0$  die  $n$ -te Catalan-Zahl  $C_n$  berechnet, wobei

$$C_n = \frac{(2n)!}{(n+1)! \cdot n!}$$

### Exkurs: Fakultät

Für alle natürlichen Zahlen  $n$  ist

$$n! = 1 \cdot 2 \cdot 3 \cdots n = \prod_{k=1}^n k$$

als das Produkt der natürlichen Zahlen von 1 bis  $n$  definiert. Da das leere Produkt stets 1 ist, gilt

$$0! = 1$$

Die Fakultät lässt sich auch rekursiv definieren:

$$n! = \begin{cases} 1, & n = 0 \\ n \cdot (n-1)!, & n > 0 \end{cases}$$

Fakultäten für negative oder nicht ganze Zahlen sind nicht definiert. Es gibt aber eine Erweiterung der Fakultät auf solche Argumente <sup>a</sup>

<sup>a</sup>[https://de.wikipedia.org/wiki/Fakultät\\_\(Mathematik\)](https://de.wikipedia.org/wiki/Fakultät_(Mathematik))

**Exkurs: Catalan-Zahl**

Die Catalan-Zahlen bilden eine Folge natürlicher Zahlen, die in vielen Problemen der Kombinatorik auftritt. Sie sind nach dem belgischen Mathematiker Eugène Charles Catalan benannt.

Die Folge der Catalan-Zahlen  $C_0, C_1, C_2, C_3, \dots$  beginnt mit  $1, 1, 2, 5, 14, 42, 132, \dots$  <sup>a</sup>

<sup>a</sup><https://de.wikipedia.org/wiki/Catalan-Zahl>

Beim Induktionsschritt können Sie die beiden folgenden Gleichungen verwenden:

- (i)  $(2(n+1))! = (4n+2) \cdot (n+1) \cdot (2n)!$
- (ii)  $(n+2)! \cdot (n+1)! = (n+2) \cdot (n+1) \cdot (n+1)! \cdot n!$

Lösungsvorschlag

**Induktionsanfang**

— Beweise, dass  $A(1)$  eine wahre Aussage ist. —

$$\begin{aligned}
 C_0 &= \frac{(2 \cdot 0)!}{(0+1)! \cdot 0!} \\
 &= \frac{0!}{1! \cdot 0!} \\
 &= \frac{1}{1 \cdot 1} \\
 &= \frac{1}{1} \\
 &= 1
 \end{aligned}$$

**Induktionsvoraussetzung**

— Die Aussage  $A(k)$  ist wahr für ein beliebiges  $k \in \mathbb{N}$ . —

$$C_n = \frac{(2n)!}{(n+1)! \cdot n!}$$

## Induktionsschritt

— Beweise, dass wenn  $A(n = k)$  wahr ist, auch  $A(n = k + 1)$  wahr sein muss.

## Vom Code ausgehend

$$\begin{aligned}
 C_{n+1} &= \frac{(4 \cdot (n+1-1) + 2) \cdot \text{cn}(n+1-1)}{n+1+1} && \text{Java nach Mathe} \\
 &= \frac{(4n+2) \cdot \text{cn}(n)}{n+2} && \text{addiert, subtrahiert} \\
 &= \frac{(4n+2) \cdot (2n)!}{(n+2) \cdot (n+1)! \cdot n!} && \text{für cn(n) Formel eingesetzt} \\
 &= \frac{(4n+2) \cdot (2n)! \cdot (n+1)}{(n+2) \cdot (n+1)! \cdot n! \cdot (n+1)} && (n+1) \text{ multipliziert} \\
 &= \frac{(4n+2) \cdot (n+1) \cdot (2n)!}{(n+2) \cdot (n+1)! \cdot (n+1) \cdot n!} && \text{umsortiert} \\
 &= \frac{(2(n+1))!}{(n+2)! \cdot (n+1)!} && \text{Hilfsgleichungen verwendet} \\
 &= \frac{(2(n+1))!}{((n+1)+1)! \cdot (n+1)!} && (n+1) \text{ verdeutlicht}
 \end{aligned}$$

## Mathematische Herangehensweise

$$\begin{aligned}
 C_{n+1} &= \frac{(2(n+1))!}{((n+1)+1)! \cdot (n+1)!} && n+1 \text{ in } C_n \text{ eingesetzt} \\
 &= \frac{(2(n+1))!}{(n+2)! \cdot (n+1)!} && \text{addiert} \\
 &= \frac{(4n+2) \cdot (n+1) \cdot (2n)!}{(n+2) \cdot (n+1) \cdot (n+1)! \cdot n!} && \text{Hilfsgleichungen verwendet} \\
 &= \frac{(4n+2) \cdot (2n)!}{(n+2) \cdot (n+1)! \cdot n!} && (n+1) \text{ gekürzt} \\
 &= \frac{4n+2}{n+2} \cdot C_n && \text{Catalan-Formel ersetzt} \\
 &= \frac{4((n+1)-1)+2}{(n+1)+1} \cdot C_{(n+1)-1} && (n+1) \text{ verdeutlicht}
 \end{aligned}$$

- (b) Geben Sie eine geeignete Terminierungsfunktion an und begründen Sie, warum der Methodenaufruf  $cn(n)$  für jedes  $n \geq 0$  terminiert.

Lösungsvorschlag

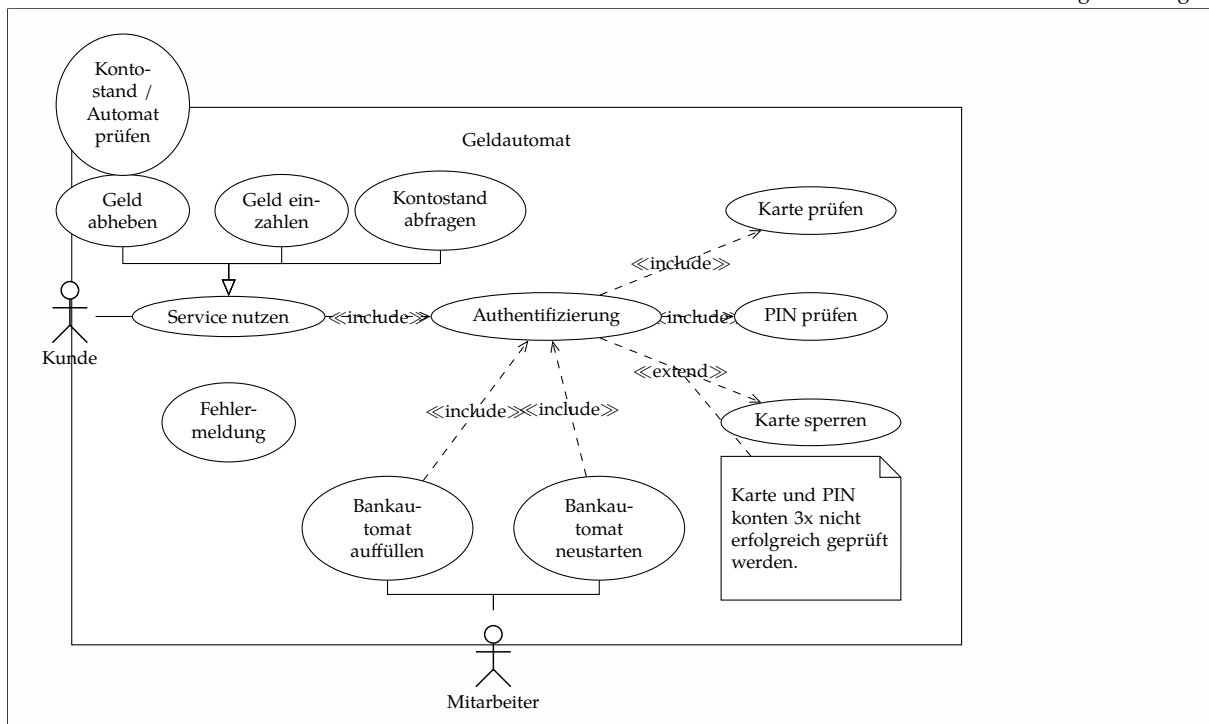
$T(n) = n$ . Diese Funktion verringert sich bei jedem Rekursionsschritt um eins. Sie ist monoton fallend und für  $T(0) = 0$  definiert. Damit ist sie eine Terminierungsfunktion für  $cn(n)$ .

## 46116 / 2017 / Frühjahr / Thema 1 / Teilaufgabe 1 / Aufgabe 2

Im Folgenden sehen Sie ein fehlerhaftes Use-Case-Diagramm für das System „Geldautomat“. Geben Sie ein korrektes Use-Case-Diagramm (Beziehungen und Beschriftungen) entsprechend der folgenden Beschreibung an. Sollte es mehrere Möglichkeiten geben, begründen Sie Ihre Entscheidung kurz.

Kunden können am Geldautomat verschiedene Services nutzen, es kann Geld abgehoben und eingezahlt sowie der Kontostand abgefragt werden. Für jeden dieser Services ist eine Authentifizierung notwendig. Diese besteht aus der Prüfung der Bankkarte und des eingegebenen PINs. Manche Bankautomaten können Karten bei zu vielen Fehlversuchen (3) bei der Anmeldung sperren, andere geben Fehlermeldungen aus, falls die Karte gesperrt wurde oder nicht mehr genügend Geld auf dem Konto oder im Bankautomaten vorhanden ist. Mitarbeiter bzw. Mitarbeiterinnen der Bank können sich ebenfalls über PIN und Karte authentifizieren, um dann den Bankautomaten neu zu starten oder mit Geld aufzufüllen. Bevor Geld abgehoben werden kann, ist sicherzustellen, dass auf dem Konto und im Bankautomaten genügend Geld vorhanden ist.

Lösungsvorschlag



**46116 / 2017 / Frühjahr / Thema 1 / Teilaufgabe 1 / Aufgabe 3**

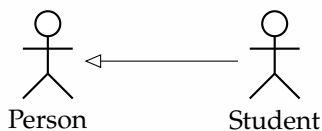
Betrachten Sie die folgenden UML-Diagramme. Sind diese korrekt? Falls nein, begründen Sie warum nicht. Geben Sie in diesem Fall außerdem eine korrigierte Version an. Falls ja, erklären Sie die inhaltliche Bedeutung des Diagramms.

(a)

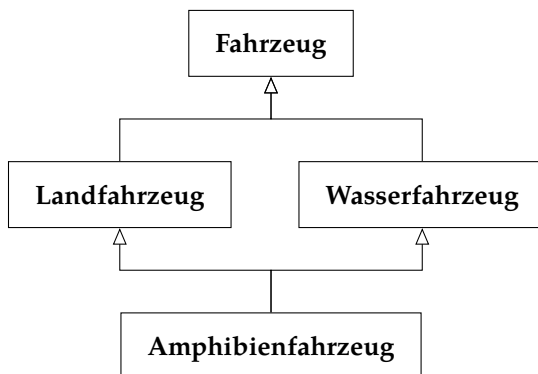


Lösungsvorschlag

*Falsch*, den verwendeten „extends“-Pfeil gibt es nur zwischen Anwendungsfällen.



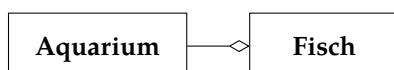
(b)



Lösungsvorschlag

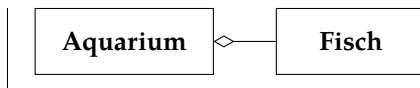
Die dargestellte Modellierung ist *korrekt*. Sowohl Land- als auch Wasserfahrzeuge sind Fahrzeuge und erben somit von dieser Klasse. Da ein Amphibienfahrzeug eine „Mischung“ aus beidem ist, erbt diese Klasse auch von beiden Klassen. Diese Mehrfachvererbung kann allerdings nicht in jeder Programmiersprache (z. B. nicht in Java) umgesetzt werden.

(c)



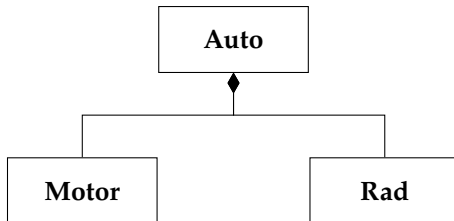
Lösungsvorschlag

Bei der dargestellten Aggregation befindet sich die Raute an der *falschen* Seite der Beziehung. Das Diagramm würde bedeuten, dass ein Fisch mehrere Aquarien enthält. Die Umkehrung ist aber korrekt:



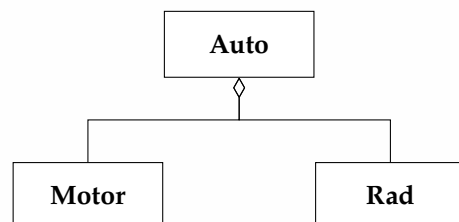
Gantt-Diagramm

(d)



Lösungsvorschlag

Hier wurde für die Modellierung eine Komposition gewählt. Dies bedeutet, dass die Existenz der Teile vom Ganzen abhängt. Einen Raum kann es z. B. ohne ein Gebäude nicht geben. In diesem Fall ist die Darstellung *falsch*, da Motor und Rad auch ohne Auto existieren können. Die Modellierung muss also mittels Aggregation erfolgen:



## 46116 / 2017 / Frühjahr / Thema 1 / Teilaufgabe 1 / Aufgabe 5

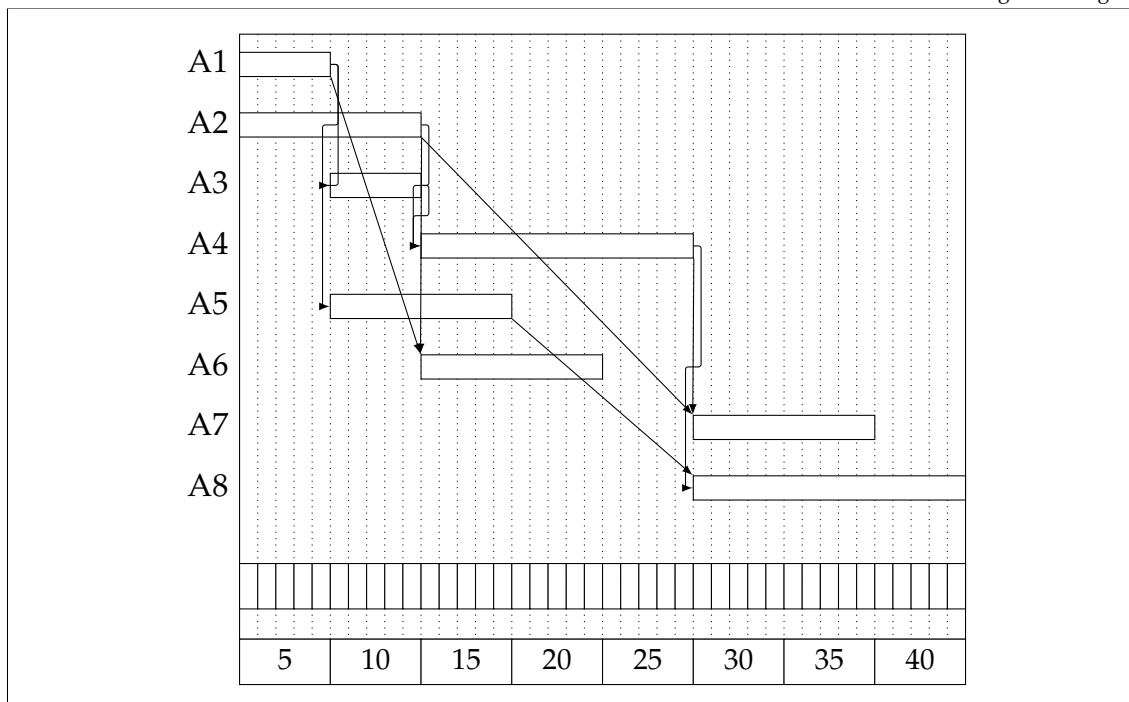
Betrachten Sie die folgende Tabelle zum Projektmanagement:

Name	Dauer (Tage)	Abhängig von
A1	5	
A2	10	
A3	5	A1
AA	15	A2, A3
AS	10	A1
A6	10	A1, A2
A7	10	A2, A4
A8	15	A4, A5

Tabelle 1: Übersicht Arbeitspakete

- (a) Erstellen Sie ein Gantt-Diagramm, das die in der Tabelle angegebenen Abhängigkeiten berücksichtigt.





(b) Wie lange dauert das Projekt mindestens?

Lösungsvorschlag

40 Tage

(c) Geben Sie den oder die kritischen Pfad(e) an.

Lösungsvorschlag

A2 A4 A8

A1 A3 A4 A8

(d) Konstruieren Sie ein PERT-Chart zum obigen Problem.

## 46116 / 2017 / Herbst / Thema 1 / Teilaufgabe 1 / Aufgabe 4

Ein gängiger Ansatz zur Messung der Qualität von Software ist das automatisierte Testen von Programmen. Im Folgenden werden praktische Testmethoden anhand des nachstehend angegebenen Sortieralgorithmus diskutiert.

### Algorithmus 1 Bubble Sort

```
public class BubbleSort {
    void bubblesort(int[] array, int len) {
        for (int i = 0; i < len - 1; i++) { // 1
            for (int j = 0; j < len - 1; j++) { // 2
                if (array[j] > array[j + 1]) { // 3
                    int temp = array[j]; // 4
                    array[j] = array[j + 1]; // 5
                    array[j + 1] = temp;
                }
            }
        }
    }
}
```

```

        array[j + 1] = temp;           // 6
    }
}
}
}
}

```

Black-Box-Testing  
Kontrollflussgraph  
C2a Vollständige  
Pfadüberdeckung (Full  
Path Coverage)

Code-Beispiel auf Github ansehen: [src/main/java/org/bschlangaul/examen/examen\\_46116/jahr\\_2017/herbst/BubbleSort.java](https://github.com/src/main/java/org/bschlangaul/examen/examen_46116/jahr_2017/herbst/BubbleSort.java)

- (a) Nennen Sie eine Art des Black-Box-Testens und beschreiben Sie deren Durchführung anhand des vorgegebenen Algorithmus.

Lösungsvorschlag

Beim Black-Box-Testen sind die Testfälle von Daten getrieben (Data-Driven) und beziehen sich auf die Anforderungen und das spezifizierte Verhalten.)

⇒ Aufruf der Methoden mit verschiedenen Eingangsparametern und Vergleich der erhaltenen Ergebnisse mit den erwarteten Ergebnissen.

Das Ziel ist dabei eine möglichst hohe Anforderungsüberdeckung, wobei man eine minimale Anzahl von Testfällen durch Äquivalenzklassenzerlegung (1) und Grenzwertanalyse (2) erhält.

**zu (1):** Man identifiziert Bereiche von Eingabewerten, die jeweils dieselben Ergebnisse liefern. Dies sind die sog. Äquivalenzklassen. Aus diesen wählt man nun je einen Repräsentanten und nutzen diesen für den Testfall.

**zu (2):** Bei der Grenzwertanalyse identifiziert man die Grenzbereiche der Eingabedaten und wählt Daten aus dem nahen Umfeld dieser für seine Testfälle.

Angewendet auf den gegebenen Bubblesort-Algorithmus würde die Grenzwertanalyse bedeuten, dass man ein bereits aufsteigend sortiertes Array und ein absteigend sortiertes Array übergibt.

- (b) Zeichnen Sie ein mit Zeilennummern beschriftetes Kontrollflussdiagramm für den oben angegebenen Sortieralgorithmus.

Lösungsvorschlag

Zur Erinnerung: Eine im Code enthaltene Wiederholung mit for muss wie folgt im Kontrollflussgraphen „zerlegt“ werden:

- (c) Erklären Sie, ob eine vollständige Pfadüberdeckung für die gegebene Funktion möglich und sinnvoll ist.

Lösungsvorschlag

Eine vollständige Pfadüberdeckung ( $C_1$ -Test) kann nicht erreicht werden, da die Bedingung der inneren Wiederholung immer wahr ist, wenn die Bedingung der äußeren Wiederholung wahr ist. D. h., der Pfad S-1-1-2-2-1“ kann nie gegangen werden. Dies wäre aber auch nicht sinnvoll, weil jeder Eintrag

mit jedem anderen verglichen werden soll und im Fall true → false ein Durchgang ausgelassen.

Entwurfsmuster  
Klassendiagramm  
Abstrakte Fabrik (Abstract  
Factory)  
Beobachter (Observer)  
Kompositum (Composite)  
Klassendiagramm

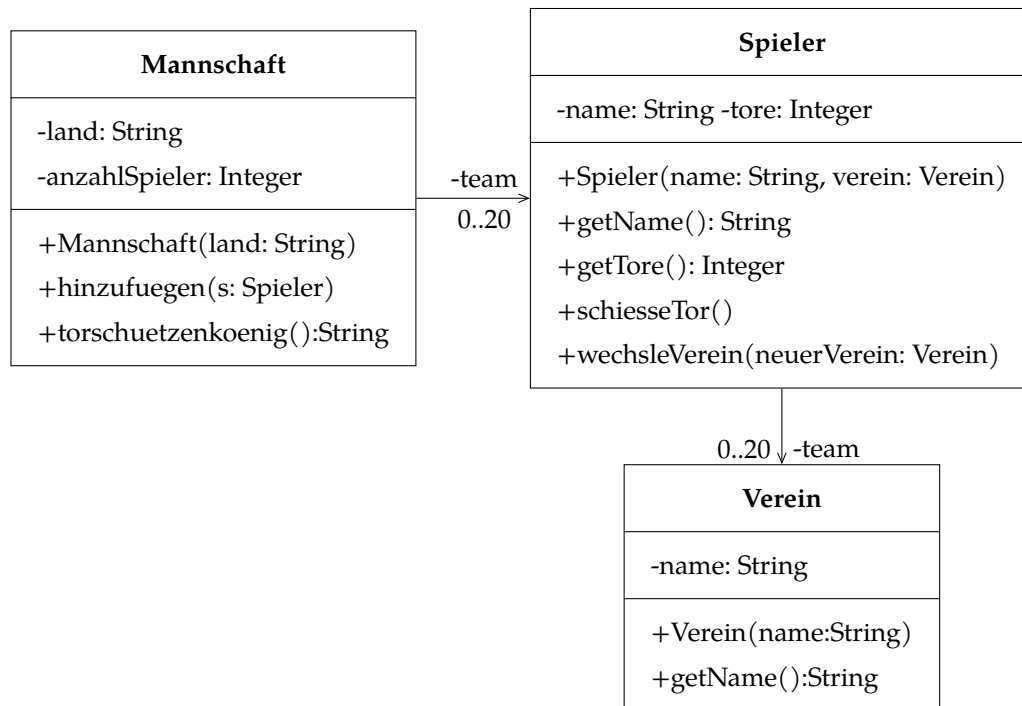
### 46116 / 2017 / Herbst / Thema 2 / Teilaufgabe 1 / Aufgabe 3

Verwenden Sie geeignete **Entwurfsmuster**, um die folgenden Sachverhalte mit Hilfe von **UML-Klassendiagrammen** zu beschreiben. Nennen Sie das zu verwendende Entwurfsmuster namentlich, wenden Sie es zur Lösung der jeweiligen *Fragestellung* an und erstellen Sie damit das problemspezifische UML-Klassendiagramm. Beschränken Sie sich dabei auf die statische Sicht, definieren Sie keinerlei Verhalten mit Ausnahme der Definition geeigneter Operationen.

- (a) Es gibt unterschiedliche Arten von Bankkonten: Girokonto, Bausparkonto vmd Kreditkarte. Bei allen Konten ist der Name des Inhabers hinterlegt. Girokonten haben eine IBAN. Kreditkarten sind immer mit einem Girokonto verknüpft. Bei Bausparkonten werden ein Sparzins sowie ein Darlehenszins festgelegt. Es gibt eine *zentrale* Klasse, die die *Erzeugung* unterschiedlicher Typen von Bankkonten steuert.
- (b) Beim Ticker für ein Hockeyspiel können sich verschiedene Geräte registrieren und wieder abmelden, um auf *Veränderungen* des Spielstands zu *reagieren*. Hierzu werden im Ticker die Tore der Heim- vmd Gastmannschaft sowie die aktuelle Spielminute vermerkt. Als konkrete Geräte sind eine Smartphone-App sowie eine Stadionuhr bekannt.
- (c) Dateisysteme sind *baumartig* strukturiert. Verzeichnisse können wiederum selbst Verzeichnisse und/oder Dateien beinhalten. Sowohl Dateien als auch Verzeichnisse haben einen Namen. Das jeweilige Elternverzeichnis ist eindeutig. Bei Dateien wird die Art (Binär, Text oder andere) sowie die Größe in Byte, bei Verzeichnissen die Anzahl enthaltener Dateien hinterlegt.

### 46116 / 2018 / Frühjahr / Thema 1 / Teilaufgabe 1 / Aufgabe 3

Für die nächste Fußballweltmeisterschaft möchte ein Wettbüro ein Programm zur Verwaltung von Spielern, Vereinen und (National-)Mannschaften entwickeln. Dazu wurde bereits das folgende UML-Klassendiagramm entworfen.



Es kann angenommen werden, dass die Klasse **Verein** bereits implementiert ist. In den folgenden Implementierungsaufgaben können Sie eine objektorientierte Programmiersprache Ihrer Wahl verwenden. Die verwendete Sprache ist anzugeben. Zu beachten sind jeweils die im Klassendiagramm angegebenen Sichtbarkeiten von Attributen, Rollennamen, Konstruktoren und Operationen.

- (a) Es ist eine Implementierung der Klasse **Spieler** anzugeben. Der Konstruktor soll die Instanzvariablen mit den gegebenen Parametern initialisieren, wobei die Anzahl der Tore nach der Objekterzeugung gleich 0 sein soll. Ansonsten kann die Funktionalität der einzelnen Operationen aus deren Namen geschlossen werden.

```
public class Spieler {  
    private String name;  
    private int tore;  
    @SuppressWarnings("unused")  
    private Verein verein;  
  
    public Spieler(String name, Verein verein) {  
        this.name = name;  
        this.verein = verein;  
        tore = 0;  
    }  
  
    public String getName() {  
        return name;  
    }  
  
    public int getTore() {  
        return tore;  
    }  
}
```

```
public void schiesseTor() {
    tore++;
}

public void wechsleVerein(Verein neuerVerein) {
    verein = neuerVerein;
}
}
```

Code-Beispiel auf Github ansehen: [src/main/java/org/bschlangaul/examen/examen\\_46116/jahr\\_2018/fruehjahr/verein/Spieler.java](https://github.com/bschlangaul/examen/examen_46116/jahr_2018/fruehjahr/verein/Spieler.java)

Feld (Array)

- (b) Es ist eine Implementierung der Klasse `Mannschaft` anzugeben. Der Konstruktor soll das Land initialisieren, die Anzahl der Spieler auf 0 setzen und das Team mit einem noch „leeren“ Array der Länge 20 initialisieren. Das Team soll mit der Methode `hinzufuegen` um einen Spieler erweitert werden. Die Methode `torschuetzenkoenig` soll den Namen eines Spielers aus dem Team zurückgeben, der die meisten Tore für die Mannschaft geschossen hat. Ist das für mehrere Spieler der Fall, dann kann der Name eines beliebigen solchen Spielers zurückgegeben werden. Ist noch kein Spieler im Team, dann soll der String `"Kein Spieler vorhanden"` zurückgegeben werden.

```
public class Mannschaft {
    @SuppressWarnings("unused")
    private String land;
    private int anzahlSpieler;
    private Spieler[] team;

    public Mannschaft(String land) {
        this.land = land;
        anzahlSpieler = 0;
        team = new Spieler[20];
    }

    public void hinzufuegen(Spieler s) {
        team[anzahlSpieler] = s;
        anzahlSpieler++;
    }

    public String torschuetzenkoenig() {
        if (anzahlSpieler == 0) {
            return "Kein Spieler vorhanden";
        }
        Spieler koenig = team[0];
        for (int i = 0; i < anzahlSpieler; i++) {
            Spieler spieler = team[i];
            if (spieler.getTore() > koenig.getTore()) {
                koenig = spieler;
            }
        }
        return koenig.getName();
    }
}
```

}

main-Methode  
Zustandsdiagramm zeichnenCode-Beispiel auf Github ansehen: [src/main/java/org/bschlangaul/examen/examen\\_46116/jahr\\_2018/fruehjahr/verein/Mannschaft.java](https://github.com/orgs/bschlangaul/repositories/examen/examen_46116/jahr_2018/fruehjahr/verein/Mannschaft.java)

- (c) Schreiben Sie den Rumpf einer `main`-Methode, so dass nach Ausführung der Methode eine deutsche Mannschaft existiert mit zwei Spielern Namens `"Hugo Meier"` und `"Frank Huber"`. Beide Spieler sollen zum selben Verein `"FC Staatsexamen"` gehören. `"Hugo Meier"` soll nach Aufnahme in die deutsche Mannschaft genau ein Tor geschossen haben, während `"Frank Huber"` noch kein Tor erzielt hat. (Wir abstrahieren hier von der Realität, in der ein 2-er Team noch gar nicht spielbereit ist.)

Lösungsvorschlag

```
public class Verein {
    private String name;

    public Verein(String name) {
        this.name = name;
    }

    public String getName() {
        return name;
    }

    public static void main(String[] args) {
        Verein verein1 = new Verein("FC Staatsexamen");
        Spieler spieler1 = new Spieler("Hugo Meier", verein1);
        Spieler spieler2 = new Spieler("Frank Huber", verein1);
        Mannschaft deutscheMannschaft = new Mannschaft("Deutschland");
        deutscheMannschaft.hinzufuegen(spieler1);
        deutscheMannschaft.hinzufuegen(spieler2);
        spieler1.schiesseTor();
    }
}
```

Code-Beispiel auf Github ansehen: [src/main/java/org/bschlangaul/examen/examen\\_46116/jahr\\_2018/fruehjahr/verein/Verein.java](https://github.com/orgs/bschlangaul/repositories/examen/examen_46116/jahr_2018/fruehjahr/verein/Verein.java)

## 46116 / 2018 / Herbst / Thema 1 / Teilaufgabe 1 / Aufgabe 3

Eine Digitaluhr kann alternativ entweder die Zeit (Stunden und Minuten) oder das Datum (Tag, Monat und Jahr) anzeigen. Zu Beginn zeigt die Uhr die Zeit an. Sie besitzt drei Druckknöpfe **A**, **B** und **C**. Mit Knopf **A** kann zwischen Zeit- und Datumsanzeige hin und her gewechselt werden.

Wird die Zeit angezeigt, dann kann mit Knopf **B** der Reihe nach erst in einen Stundenmodus, dann in einen Minutenmodus und schließlich zurück zur Zeitanzeige gewechselt werden. Im Stundenmodus blinkt die Stundenanzeige. Mit Drücken des Knopfes **C** können dann die Stunden schrittweise inkrementiert werden. Im Minutenmodus blinkt die Minutenanzeige und es können mit Hilfe des Knopfes **C** die Minuten schrittweise inkrementiert werden.



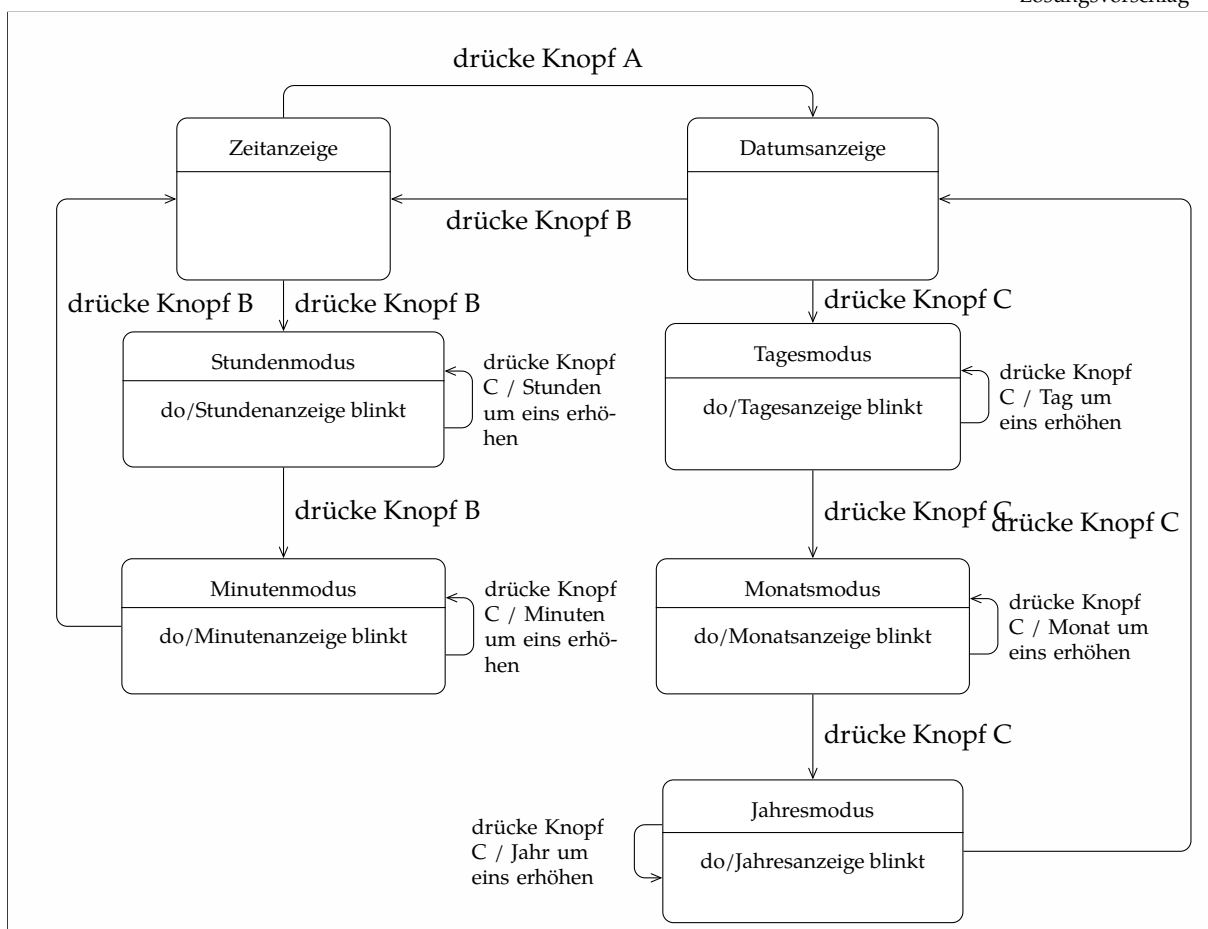
Die Datumsfunktionen sind analog. Wird das Datum angezeigt, dann kann mit Knopf **B** der Reihe nach in einen Tagesmodus, Monatsmodus, Jahresmodus und schließlich zurück zur Datumsanzeige gewechselt werden. Im Tagesmodus blinkt die Tagesanzeige. Mit Drücken des Knopfes **C** können dann die Tage schrittweise inkrementiert werden. Analog blinken mit Eintritt in den entsprechenden Einstellmodus der Monat oder das Jahr, die dann mit Knopf **C** schrittweise inkrementiert werden können.

Wenn sich die Uhr in einem Einstellmodus befindet, hat das Betätigen des Knopfes **A** keine Wirkung. Ebenso wirkungslos ist Knopf **C**, wenn gerade Zeit oder Datum angezeigt wird.

Beschreiben Sie das Verhalten der Digitaluhr durch ein UML-Zustandsdiagramm. Dabei muss - gemäß der UML-Notation - unterscheidbar sein, was Ereignisse und was Aktionen sind. Deren Bedeutung soll durch die Verwendung von sprechenden Namen klar sein. Für die Inkrementierung von Stunden, Minuten, Tagen etc. brauchen keine konkreten Berechnungen angegeben werden. Der kontinuierliche Zeitfortschritt des Uhrwerks ist nicht zu modellieren.

Zustände sind, wie in der UML üblich, durch abgerundete Rechtecke darzustellen. Sie können unterteilt werden in eine obere und eine untere Hälfte, wobei der Name des Zustands in den oberen Teil und eine in dem Zustand auszuführende Aktivität in den unteren Teil einzutragen ist.

Lösungsvorschlag



**66112 / 2002 / Herbst / Thema 1 / Aufgabe 4**

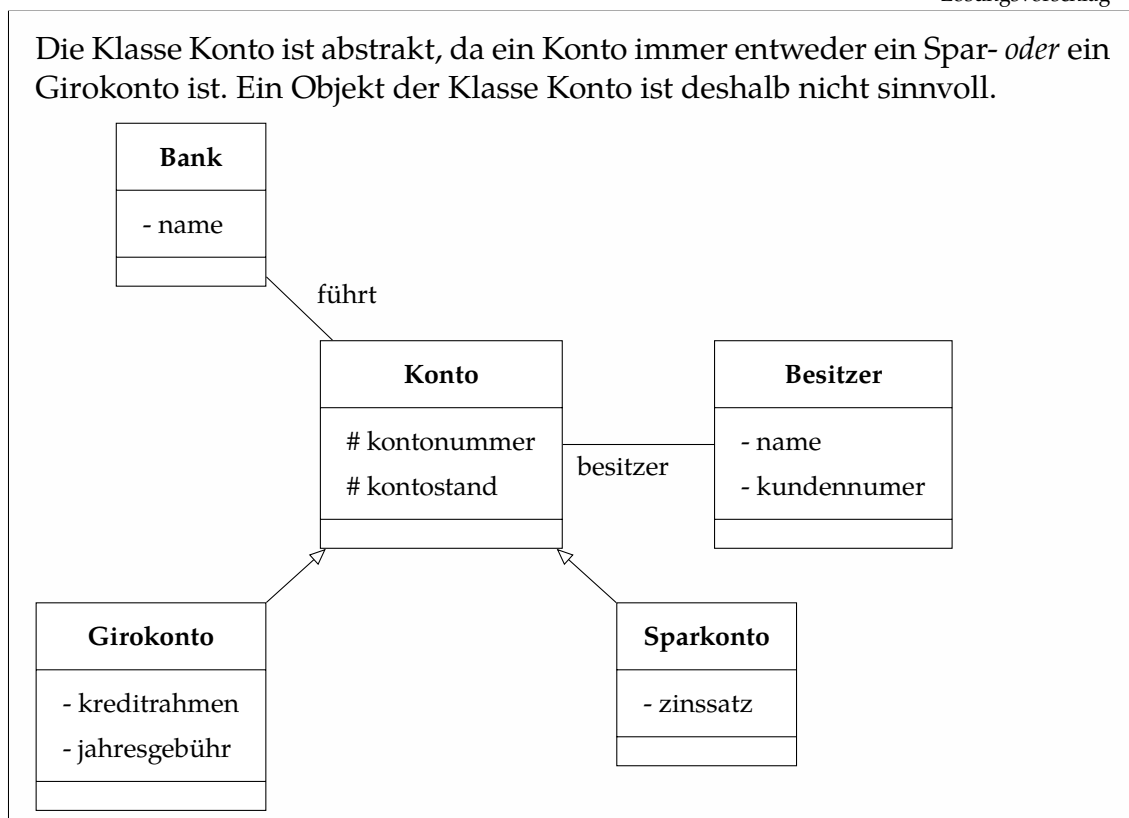
In einer Anforderungsanalyse für ein Banksystem wird der folgende Sachverhalt beschrieben:

Eine Bank hat einen Namen und sie führt Konten. Jedes Konto hat eine Kontonummer, einen Kontostand und einen Besitzer. Der Besitzer hat einen Namen und eine Kundennummer. Ein Konto ist entweder ein Sparkonto oder ein Girokonto. Ein Sparkonto hat einen Zinssatz, ein Girokonto hat einen Kreditrahmen und eine Jahresgebühr.

- (a) Deklarieren Sie geeignete Klassen in Java [oder in C++], die die oben beschriebenen Anforderungen widerspiegeln! Nutzen Sie dabei das Vererbungskonzept aus, wo es sinnvoll ist! Gibt es Klassen, die als abstrakt zu verstehen sind?

[Hinweis: Geben Sie sowohl ein Klassendiagramm als auch den Quellcode für die beteiligten Klassen inkl. Attributen, ohne Methoden an! Achtung: in Teilaufgabe b) werden anschließend die benötigten Konstruktoren verlangt; Um die verschiedenen Konten in einer Bank zu verwalten, eignet sich das Array NICHT als Datenstruktur. Informieren Sie sich über die Datenstruktur „ArrayList“ und verwenden Sie diese.]

Lösungsvorschlag



- (b) Geben Sie für alle nicht abstrakten Klassen benutzerdefinierte Konstruktoren an mit Parametern zur Initialisierung der folgenden Werte: der Name einer Bank, die Kontonummer, der Kontostand, der Besitzer und der Zinssatz (bzw. Kreditrahmen und Jahresgebühr) eines Sparkontos (bzw. Girokontos), der Name und die Kundennummer eines Kontobesitzers.

Ergänzen Sie die Klassen um Methoden für die folgenden Aufgaben! Nutzen Sie wann immer möglich das Vererbungskonzept aus und verwenden Sie ggf. abstrakte (bzw. virtuelle) Methoden!

[Achtung: Ergänzen Sie ggf. alle benötigten Getter und Setter in den beteiligten Klassen!]

Lösungsvorschlag

Konstrukturen ergänzen:

Bemerkung: Auch eine abstrakte Klasse kann einen Konstruktor besitzen, dieser kann nur nicht ausgeführt werden. In den abgeleiteten Klassen kann dieser Super-Konstruktor aber verwendet werden.

- (c) Auf ein Konto soll ein Betrag eingezahlt und ein Betrag abgehoben werden können. Soll von einem Sparkonto ein Betrag abgehoben werden, dann darf der Kontostand nicht negativ werden. Bei einer Abhebung von einem Girokonto darf der Kreditrahmen nicht überzogen werden.

Lösungsvorschlag

Bemerkung: Die Methode `einzahlen` ist in der Klasse `Konto` implementiert, da sie sich für Spar- und Girokonten nicht unterscheidet im Gegensatz zur Methode `abheben`, die in beiden Klassen unterschiedlich implementiert ist. [Sie wird als abstrakte Methode in der Klasse `Konto` angegeben, um ihre Implementierung (Überschreiben) zu gewährleisten.] Kreditrahmen wird als negativer Wert gespeichert. Die Methoden zum Abheben liefern zusätzlich zur Änderung des Kontostandes eine Rückmeldung bezüglich Erfolg oder Misserfolg der Abbuchung.

- (d) Ein Konto kann eine Jahresabrechnung durchführen. Bei der Jahresabrechnung eines Sparkontos wird der Zinsertrag gutgeschrieben, bei der Jahresabrechnung eines Girokontos wird die Jahresgebühr abgezogen (auch wenn dadurch der Kreditrahmen überzogen wird).

Lösungsvorschlag

Anmerkung: Im Folgenden nur noch Angabe der gesuchten Methoden, alle vorherigen Implementierungen (Attribute, Konstruktoren, Methoden, s. o.) wurden nicht nochmals aufgeführt.

- (e) Eine Bank kann einen Jahresabschluss durchführen. Dieser bewirkt, dass für jedes Konto der Bank eine Jahresabrechnung durchgeführt wird.
- (f) Eine Bank kann ein Sparkonto eröffnen. Die Methode soll die folgenden fünf Parameter haben: den Namen und die Kundennummer des Kontobesitzers, die Kontonummer, den (anfänglichen) Kontostand und den Zinssatz des Sparkontos. Alle Parameter sind als String-Objekte oder als Werte eines Grunddatentyps zu übergeben! Das Sparkonto muss nach seiner Eröffnung in den Kontenbestand der Bank aufgenommen sein.

[Hinweis: Falls der Kunde schon mit einem Konto in der Bank geführt ist, können die Werte für das `Besitzer`-Objekt übernommen werden. Schreiben Sie daher eine Hilfsmethode `Besitzer schonVorhanden(String name, int kunr)`, die prüft, ob der Kunde mit `name` und `kunr` schon in der Liste vorhanden ist und diesen bzw. andernfalls einen neu erzeugten Besitzer zurückgibt.]

[Sinnvolle/notwendige Methoden der Klasse `ArrayList` für diese Aufgabe:

`public int size()` : Returns the number of elements in this list.

`public boolean isEmpty()` : Returns true if this list contains no elements.

`public E get(int index)` : Returns the element at the specified position in this list.

`public boolean add(E e)` : Appends the specified element to the end of this list.

(siehe auch Java-API: <https://docs.oracle.com/javase/7/docs/api/java/util/ArrayList.html>)]

Lösungsvorschlag

```
import java.util.ArrayList;

public class Bank {
    @SuppressWarnings("unused")
    private String name;
    private ArrayList<Konto> konten;

    public Bank(String name) {
        this.name = name;
        konten = new ArrayList<Konto>();
    }

    public void rechneAb() {
        for (int i = 0; i <= konten.size(); i++) {
            konten.get(i).rechneAb();
        }
    }

    public void legeAn(String name, int kundenNummer, int kontoNummer, float
→ kontoStand, float zinssatz) {
        Besitzer besitzer = schonVorhanden(name, kundenNummer);
        konten.add(new Sparkonto(kontoNummer, kontoStand, besitzer, zinssatz));
    }

    public Besitzer schonVorhanden(String name, int kundenNummer) {
        if (!konten.isEmpty()) {
            for (int i = 0; i < konten.size(); i++) {
                if (konten.get(i).besitzer.gibKundenNummer() == kundenNummer) {
                    return konten.get(i).gibBesitzer();
                }
            }
        }
        return new Besitzer(name, kundenNummer);
    }
}

Code-Beispiel auf Github ansehen: src/main/java/org/bschlangaul/examen/examen_66112/jahr_2002/herbst/Bank.java

public abstract class Konto {
```

```
protected int kontoNummer;
protected float kontoStand;
protected Besitzer besitzer;

public Konto(int kontoNummer, float kontoStand, Besitzer besitzer) {
    this.kontoNummer = kontoNummer;
    this.kontoStand = kontoStand;
    this.besitzer = besitzer;
}

public void zahleEin(float betrag) {
    kontoStand += betrag;
}

public Besitzer gibBesitzer() {
    return besitzer;
}

public abstract boolean hebeAb(float betrag);

public abstract void rechneAb();
}
```

Code-Beispiel auf Github ansehen: [src/main/java/org/bschlangaul/examen/examen\\_66112/jahr\\_2002/herbst/Konto.java](https://github.com/bschlangaul/examen/examen_66112/jahr_2002/herbst/Konto.java)

```
public class Sparkonto extends Konto {
    private float zinssatz;

    public Sparkonto(int kontoNummer, float kontoStand, Besitzer besitzer,
        ↪ float zinssatz) {
        super(kontoNummer, kontoStand, besitzer);
        this.zinssatz = zinssatz;
    }

    public boolean hebeAb(float betrag) {
        if (kontoStand - betrag >= 0) {
            kontoStand -= betrag;
            return true;
        } else {
            return false;
        }
    }

    public void rechneAb() {
        kontoStand += kontoStand * (1 + zinssatz);
    }
}
```

Code-Beispiel auf Github ansehen: [src/main/java/org/bschlangaul/examen/examen\\_66112/jahr\\_2002/herbst/Sparkonto.java](https://github.com/bschlangaul/examen/examen_66112/jahr_2002/herbst/Sparkonto.java)

```
public class Girokonto extends Konto {
    private float kreditRahmen;
    private float jahresGebühr;
}
```

```

public Girokonto(int kontoNummer, float kontoStand, Besitzer besitzer,
    ↪ float kreditRahmen, float jahresGebühr) {
    super(kontoNummer, kontoStand, besitzer);
    this.kreditRahmen = kreditRahmen;
    this.jahresGebühr = jahresGebühr;
}

public boolean hebeAb(float betrag) {
    if (kontoStand - betrag >= kreditRahmen) {
        kontoStand -= betrag;
        return true;
    } else {
        return false;
    }
}

public void rechneAb() {
    kontoStand -= jahresGebühr;
}
}

```

Code-Beispiel auf Github ansehen: [src/main/java/org/bschlangaul/examen/examen\\_66112/jahr\\_2002/herbst/Girokonto.java](https://github.com/bschlangaul/examen/examen_66112/jahr_2002/herbst/Girokonto.java)

```

@SuppressWarnings("unused")
public class Besitzer {
    private String name;
    private int kundenNummer;
    private Konto hatKonto;

    public Besitzer(String name, int kundenNummer) {
        this.name = name;
        this.kundenNummer = kundenNummer;
    }

    public int gibKundenNummer() {
        return kundenNummer;
    }
}

```

Code-Beispiel auf Github ansehen: [src/main/java/org/bschlangaul/examen/examen\\_66112/jahr\\_2002/herbst/Besitzer.java](https://github.com/bschlangaul/examen/examen_66112/jahr_2002/herbst/Besitzer.java)

## 66112 / 2003 / Herbst / Thema 2 / Aufgabe 5

Zeigen Sie mit Hilfe vollständiger Induktion, dass das folgende Programm bzgl. der Vorbedingung  $x > 0$  und der Nachbedingung  $\text{drei\_hoch } x = 3^x$  partiell korrekt ist!

```

(define (drei_hoch x)
  (cond ((= x 0) 1)
        (else (* 3 (drei_hoch (- x 1)))))
)
)

```

**Induktionsanfang**

- Beweise, dass  $A(1)$  eine wahre Aussage ist. \_\_\_\_\_  
 $\text{drei\_hoch } 1 = 3 \cdot (\text{drei\_hoch } 0) = 3 \cdot 1 = 3$

**Induktionsvoraussetzung**

- Die Aussage  $A(k)$  ist wahr für ein beliebiges  $k \in \mathbb{N}$ . \_\_\_\_\_  
 für alle  $x < x_0$  gilt  $\text{drei\_hoch } x = 3^x$

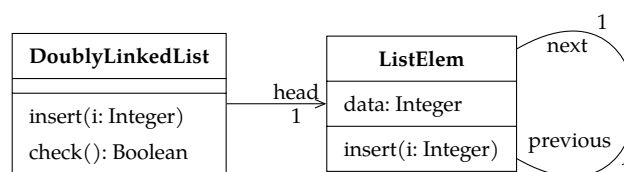
**Induktionsschritt**

- Beweise, dass wenn  $A(n = k)$  wahr ist, auch  $A(n = k + 1)$  wahr sein muss. \_\_\_\_\_  
 $x \rightarrow x+1$

$$\begin{aligned}
 \text{drei\_hoch } (x + 1) &= 3 \cdot \text{drei\_hoch } ((x + 1) - 1) \\
 &= 3 \cdot (\text{drei\_hoch } x) \\
 &= 3 \cdot 3^x \\
 &= 3^{x+1}
 \end{aligned}$$

**66112 / 2005 / Frühjahr / Thema 1 / Aufgabe 1**

Betrachten Sie folgendes Klassendiagramm, das doppelt-verkettete Listen spezifiziert. Die Assoziation **head** zeigt auf das erste Element der Liste. Die Assoziationen **previous** und **next** zeigen auf das vorherige bzw. folgende Element.



Implementieren Sie die doppelt-verketteten Listen in einer geeigneten objektorientierten Sprache (z. B. Java oder C++), das heißt:

- (a) Implementieren Sie die Klasse **ListElem**. Die Methode **insert** ordnet eine ganze Zahl **i** in eine aufsteigend geordnete doppelt-verkettete Liste **l** an die korrekte Stelle ein. Sei z. B. das Objekt **l** eine Repräsentation der Liste **[0, 2, 2, 6, 8]** dann liefert **l.insert(3)** eine Repräsentation der Liste **[0, 2, 2, 3, 6, 8]**.

```
public class ListElem {
    private int data;
    private ListElem previous;
    private ListElem next;

    public ListElem(int i) {
        data = i;
    }

    public ListElem() {
    }

    public void insert(int i) {
        ListElem newElement = new ListElem(i);
        if (i <= data) {
            if (previous != null) {
                newElement.next = this;
                newElement.previous = previous;
                previous.next = newElement;
                previous = newElement;
            } else {
                newElement.next = this;
                previous = newElement;
            }
        } else {
            if (next != null) {
                next.insert(i);
            } else {
                newElement.previous = this;
                next = newElement;
            }
        }
    }

    public ListElem getPrevious() {
        return previous;
    }

    public ListElem getNext() {
        return next;
    }

    public int getData() {
        return data;
    }
}
```

Code-Beispiel auf Github ansehen: [src/main/java/org/bschlangaul/examen/examen\\_66112/jahr\\_2005/fruehjahr/ListElem.java](https://github.com/src/main/java/org/bschlangaul/examen/examen_66112/jahr_2005/fruehjahr/ListElem.java)

- (b) Implementieren Sie die Klasse `DoublyLinkedList`, wobei die Methode `insert` eine Zahl `i` in eine aufsteigend geordnete Liste einordnet. Die Methode `check` überprüft, ob eine Liste korrekt verkettet ist, d.h. für jedes `ListElem`-Objekt `o`, das über den `head` der Liste erreichbar ist, der Vorgänger des Nachfolgers von `o`



gleich 0 ist.

Lösungsvorschlag

```
public class DoublyLinkedList {
    private ListElem head;

    public DoublyLinkedList() {
    }

    public void insert(int i) {
        if (head != null) {
            // Immer einen neue Zahl einfügen, nicht nur wenn die Zahl kleiner ist als
            ↪ head.
            head.insert(i);
            // Es muss kleiner gleich heißen, sonst können mehrer gleiche Zahlen am
            ↪ Anfang
            // nicht eingefügt werden.
            if (i <= head.getData()) {
                head = head.getPrevious();
            }
        } else {
            head = new ListElem(i);
        }
    }

    public boolean check() {
        ListElem current = head;
        while (current.getNext() != null) {
            if (current.getNext().getPrevious() != current) {
                return false;
            } else {
                current = current.getNext();
            }
        }
        return true;
    }

    public ListElem getHead() {
        return head;
    }

    public static void main(String[] args) {
        DoublyLinkedList list = new DoublyLinkedList();
        // int[] numbers = new int[] { 1 };
        // int[] numbers = new int[] { 1, 1, 1, 1, };
        // int[] numbers = new int[] { 1, 1, 1, 2, };
        // int[] numbers = new int[] { 2, 1, 1, 1, };
        // int[] numbers = new int[] { 2, 1 };
        int[] numbers = new int[] { 0, 2, 2, 6, 8, 4 };
        for (int number : numbers) {
            list.insert(number);
        }
        list.insert(3);
    }
}
```

```

ListElem current = list.getHead();
while (current.getNext() != null) {
    System.out.println(current.getData());
    current = current.getNext();
}
}
}

```

Code-Beispiel auf Github ansehen: [src/main/java/org/bschlangaul/examen/examen\\_66112/jahr\\_2005/fruehjahr/DoublyLinkedList.java](https://github.com/bschlangaul/examen/examen_66112/jahr_2005/fruehjahr/DoublyLinkedList.java)

## 66115 / 2014 / Frühjahr / Thema 1 / Aufgabe 1

- (a) Gegeben sei die Methode `BigInteger lfBig(int n)` zur Berechnung der eingeschränkten Linksfakultät:

$$!n := \begin{cases} n!(n-1) - (n-1)!(n-2) & \text{falls } 1 < n < 32767 \\ 1 & \text{falls } n = 1 \\ 0 & \text{sonst} \end{cases}$$

```

import java.math.BigInteger;
import static java.math.BigInteger.ZERO;
import static java.math.BigInteger.ONE;

public class LeftFactorial {

    BigInteger sub(BigInteger a, BigInteger b) {
        return a.subtract(b);
    }

    BigInteger mul(BigInteger a, BigInteger b) {
        return a.multiply(b);
    }

    BigInteger mul(int a, BigInteger b) {
        return mul(BigInteger.valueOf(a), b);
    }

    // returns the left factorial !n
    BigInteger lfBig(int n) {
        if (n <= 0 || n >= Short.MAX_VALUE) {
            return ZERO;
        } else if (n == 1) {
            return ONE;
        } else {
            return sub(mul(n, lfBig(n - 1)), mul(n - 1, lfBig(n - 2)));
        }
    }
}

```

Code-Beispiel auf Github ansehen: [src/main/java/org/bschlangaul/examen/examen\\_66115/jahr\\_2014/fruehjahr/LeftFactorial.java](https://github.com/bschlangaul/examen/examen_66115/jahr_2014/fruehjahr/LeftFactorial.java)

Implementieren Sie unter Verwendung des Konzeptes der *dynamischen Programmierung* die Methode `BigInteger dp(int n)`, die jede  $!n$  auch bei mehrfachem

Aufrufen mit dem gleichen Parameter höchstens einmal rekursiv berechnet. Sie dürfen der Klasse `LeftFactorial` genau ein Attribut beliebigen Datentyps hinzufügen und die in `lfBig(int)` verwendeten Methoden und Konstanten ebenfalls nutzen.

Lösungsvorschlag

Wir führen ein Attribut mit dem Namen `store` ein und erzeugen ein Feld vom Typ `BigInteger` mit der Länge  $n + 1$ . Die Länge des Feld  $n + 1$  hat den Vorteil, dass nicht ständig  $n - 1$  verwendet werden muss, um den gewünschten Wert zu erhalten.

In der untenstehenden Implementation gibt es zwei Methoden mit dem Namen `dp`. Die untenstehende Methode ist nur eine Hüllmethode, mit der nach außen hin die Berechnung gestartet und das `store`-Feld neu gesetzt wird. So ist es möglich `dp()` mehrmals hintereinander mit verschiedenen Werten aufzurufen (siehe `main()`-Methode).

```

BigInteger[] store;

BigInteger dp(int n, BigInteger[] store) {
    if (n > 1 && store[n] != null) {
        return store[n];
    }
    if (n <= 0 || n >= Short.MAX_VALUE) {
        return ZERO;
    } else if (n == 1) {
        return ONE;
    } else {
        BigInteger result = sub(mul(n, dp(n - 1, store)), mul(n - 1, dp(n - 2,
↪ store)));
        store[n] = result;
        return result;
    }
}

BigInteger dp(int n) {
    store = new BigInteger[n + 1];
    return dp(n, store);
}

```

Code-Beispiel auf Github ansehen: [src/main/java/org/bschlangaul/examen/examen\\_66115/jahr\\_2014/fruehjahr/LeftFactorial.java](https://github.com/bschlangaul/examen/examen_66115/jahr_2014/fruehjahr/LeftFactorial.java)

- (b) Betrachten Sie nun die Methode `lfLong(int)` zur Berechnung der vorangehend definierten Linksfakultät ohne obere Schranke. Nehmen Sie im Folgenden an, dass der Datentyp `long` unbeschränkt ist und daher kein Überlauf auftritt.

```

long lfLong(int n) {
    if (n <= 0) {
        return 0;
    } else if (n == 1) {
        return 1;
    } else {

```

```
    return n * lfLong(n - 1) - (n - 1) * lfLong(n - 2);  
  }  
}
```

Code-Beispiel auf Github ansehen: [src/main/java/org/bschlangaul/examen/examen\\_66115/jahr\\_2014/fruehjahr/LeftFactorial.java](https://github.com/bschlangaul/examen/examen_66115/jahr_2014/fruehjahr/LeftFactorial.java)

Beweisen Sie *formal* mittels *vollständiger Induktion*:

$$\forall n \geq 0 : \text{lfLong}(n) \equiv \sum_{k=0}^{n-1} k!$$

Lösungsvorschlag

### Induktionsanfang

— Beweise, dass  $A(1)$  eine wahre Aussage ist. —————

$$n = 1 \Rightarrow \text{lfLong}(1) = 1 = \sum_{k=0}^{n-1} k! = 0! = 1$$

$$n = 2 \Rightarrow \text{lfLong}(2)$$

$$\begin{aligned} &= (n + 1) \sum_{k=0}^{n-1} k! - n \sum_{k=0}^{n-2} k! \\ &= 2 * \text{lfLong}(1) - 1 * \text{lfLong}(0) \\ &= 2 \\ &= \sum_{k=0}^1 k! \\ &= 1! + 0! \\ &= 1 + 1 \\ &= 2 \end{aligned}$$

### Induktionsvoraussetzung

— Die Aussage  $A(k)$  ist wahr für ein beliebiges  $k \in \mathbb{N}$ . —————

$$\text{lfLong}(n) = \sum_{k=0}^{n-1} k!$$

### Induktionsschritt

— Beweise, dass wenn  $A(n = k)$  wahr ist, auch  $A(n = k + 1)$  wahr sein muss.

$$\begin{aligned}
 A(n+1) &= \text{lfLong}(n+1) \\
 &= (n+1) * \text{lfLong}(n) - n * \text{lfLong}(n-1) \\
 &= (n+1) \sum_{k=0}^{n-1} k! - n \sum_{k=0}^{(n-1)-1} k! && \text{Formel eingesetzt} \\
 &= (n+1) \sum_{k=0}^{n-1} k! - n \sum_{k=0}^{n-2} k! && \text{subtrahiert} \\
 &= n \sum_{k=0}^{n-1} k! + \sum_{k=0}^{n-1} k! - n \sum_{k=0}^{n-2} k! && \text{ausmultipliziert mit } (n+1) \\
 &= \sum_{k=0}^{n-1} k! + n \sum_{k=0}^{n-1} k! - n \sum_{k=0}^{n-2} k! && \text{Reihenfolge der Terme geändert} \\
 &= \sum_{k=0}^{n-1} k! + n \left( (n-1)! + \sum_{k=0}^{n-2} k! \right) - n \sum_{k=0}^{n-2} k! && (n-1)! \text{ aus Summenzeichen entfernt} \\
 &= \sum_{k=0}^{n-1} k! + n \left( (n-1)! + \sum_{k=0}^{n-2} k! - \sum_{k=0}^{n-2} k! \right) && \text{Distributivgesetz } ac - bc = (a-b)c \\
 &= \sum_{k=0}^{n-1} k! + n(n-1)! && +\Sigma - \Sigma = 0 \\
 &= \sum_{k=0}^{n-1} k! + n! && \text{Fakultät erhöht} \\
 &= \sum_{k=0}^n k! && \text{Element zum Summenzeichen hinzugefügt} \\
 &= \sum_{k=0}^{(n+1)-1} k! && \text{mit } (n+1) \text{ an der Stelle von } n
 \end{aligned}$$

## 66115 / 2017 / Frühjahr / Thema 1 / Aufgabe 4

Sie dürfen im Folgenden davon ausgehen, dass keinerlei Under- oder Overflows auftreten.

Gegeben sei folgende rekursive Methode für  $n \geq 0$ :

```

long sumOfSquares (long n) {
    if (n == 0)
        return 0;
}

```

```
else  
    return n * n + sumOfSquares(n - 1);  
}
```

(a) Beweisen Sie formal mittels vollständiger Induktion:

$$\forall n \in \mathbb{N} : \text{sumOfSquares}(n) = \frac{n(n+1)(2n+1)}{6}$$

Lösungsvorschlag

Sei  $f(n) : \frac{n(n+1)(2n+1)}{6}$

**Induktionsanfang**

— Beweise, dass  $A(1)$  eine wahre Aussage ist. \_\_\_\_\_

Für  $n = 0$  gilt:

$$\text{sumOfSquares}(0) \stackrel{\text{if}}{=} 0 = f(0)$$

**Induktionsvoraussetzung**

— Die Aussage  $A(k)$  ist wahr für ein beliebiges  $k \in \mathbb{N}$ . \_\_\_\_\_

Für ein festes  $n \in \mathbb{N}$  gelte:

$$\text{sumOfSquares}(n) = f(n)$$

**Induktionsschritt**

— Beweise, dass wenn  $A(n = k)$  wahr ist, auch  $A(n = k + 1)$  wahr sein muss.

$$n \rightarrow n + 1$$

$f(n+1) = \text{sumOfSquares}(n+1)$	Java-Methode eingesetzt
$\stackrel{\text{else}}{=} (n+1) * (n+1) + \text{sumOfSquares}(n)$	Java-Code der else-Verzweigung verwendet
$\stackrel{\text{I.H.}}{=} (n+1)(n+1) + f(n)$	mathematisch notiert
$= (n+1)(n+1) + \frac{n(n+1)(2n+1)}{6}$	Formel eingesetzt
$= (n+1)^2 + \frac{n(n+1)(2n+1)}{6}$	potenziert
$= \frac{6(n+1)^2}{6} + \frac{n(n+1)(2n+1)}{6}$	$(n+1)^2$ in Bruch umgewandelt
$= \frac{6(n+1)^2 + n(n+1)(2n+1)}{6}$	Addition gleichnamiger Brüche
$= \frac{(n+1)6(n+1) + (n+1)n(2n+1)}{6}$	$n+1$ ausklammern vorbereitet
$= \frac{(n+1)(6(n+1) + n(2n+1))}{6}$	$n+1$ ausgeklammert
$= \frac{(n+1)(6n+6+2n^2+n)}{6}$	Klammern ausmultipliziert / aufgelöst
$= \frac{(n+1)(2n^2+7n+6)}{6}$	umsortiert, addiert $6n+n=7n$
$= \frac{(n+1)(2n^2+3n+4n+6)}{6}$	Ausklammern vorbereitet
$= \frac{(n+1)(n+2)(2n+3)}{6}$	$(n+2)$ ausgeklammert
$= \frac{(n+1)((n+1)+1)(2(n+1)+1)}{6}$	$(n+1)$ verdeutlicht

a

<sup>a</sup>[https://mathcs.org/analysis/reals/infinity/answers/sm\\_sq\\_cb.html](https://mathcs.org/analysis/reals/infinity/answers/sm_sq_cb.html)

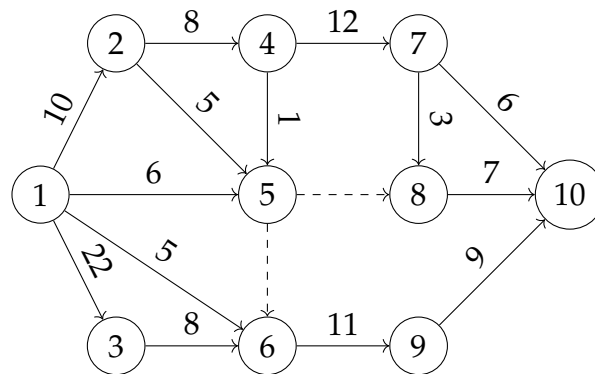
- (b) Beweisen Sie die Terminierung von `sumOfSquares(n)` für alle  $n \geq 0$ .

Lösungsvorschlag

Sei  $T(n) = n$ . Die Funktion  $T(n)$  ist offenbar ganzzahlig. In jedem Rekursionsschritt wird  $n$  um eins verringert, somit ist  $T(n)$  streng monoton fallend. Durch die Abbruchbedingung `n==0` ist  $T(n)$  insbesondere nach unten beschränkt. Somit ist  $T$  eine gültige Terminierungsfunktion.

## 66116 / 2012 / Herbst / Thema 2 / Teilaufgabe 2 / Aufgabe 2

Die unten stehende Abbildung stellt ein CPM-Netzwerk dar. Die Ereignisse sind fortlaufend nummeriert (Nummer im Inneren der Kreise) und tragen keine Namen. Gestrichelte Linien stellen Pseudo-Aktivitäten mit einer Dauer von 0 dar.



- (a) Berechnen Sie die früheste Zeit für jedes Ereignis, wobei angenommen wird, dass das Projekt zum Zeitpunkt 0 startet!

Lösungsvorschlag

— Wir führen eine Vorwärtsterminierung durch und addieren die Dauern. Kann ein Ereignis über mehrere Vorgänge erreicht werden, wählen wir das Maximum aus. **Erläuterungen:**  $i$ : Ereignis  $i$ ;  $FZ_i$ : Frühester Zeitpunkt, zu dem Ereignis  $i$  eintreten kann. —

$i$	Nebenrechnung	$FZ_i$
1		0
2		10
3		22
4		18
5	$\max(15_2, 6_1, 19_4)$	19
6	$\max(5_1, 30_6, 19_5)$	30
7		30
8	$\max(33_7, 19_5)$	33
9		41
10	$\max(36_7, 40_8, 50_9)$	50

- (b) Setzen Sie anschließend beim letzten Ereignis die späteste Zeit gleich der frühesten Zeit und berechnen Sie die spätesten Zeiten!

Lösungsvorschlag

— Wir führen eine Rückwärtsterminierung durch und subtrahieren die Dauern vom letzten Ereignis aus. Kann ein Ereignis über mehrere Vorgänge erreicht werden, wählen wir das Minimum aus. **Erläuterungen:**  $i$ : Ereignis  $i$ ;  $SZ_i$ : Spätester Zeitpunkt, zu dem Ereignis  $i$  eintreten kann. —



$i$	Nebenrechnung	$SZ_i$
10	siehe $FZ_{10}$	50
9		41
8		43
7	$\min(44_{10}, 40_8)$	40
6		30
5	$\min(30_6, 43_8)$	30
4	$\min(29_5, 28_7)$	28
3		22
2	$\min(20_4, 25_5)$	20
1	$\min(10_2, 24_5, 0_3, 25_6)$	0

(c) Berechnen Sie nun für jedes Ereignis die Pufferzeiten!

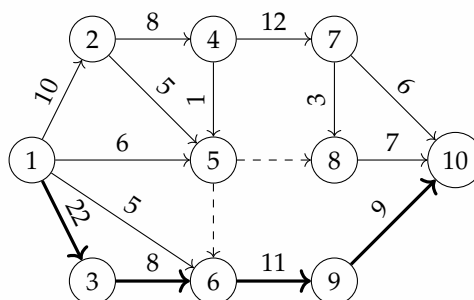
Lösungsvorschlag

$i$	1	2	3	4	5	6	7	8	9	10
$FZ_i$	0	10	22	18	19	30	30	33	41	50
$SZ_i$	0	20	22	28	30	30	40	43	41	50
GP	0	10	0	10	11	0	10	10	0	0

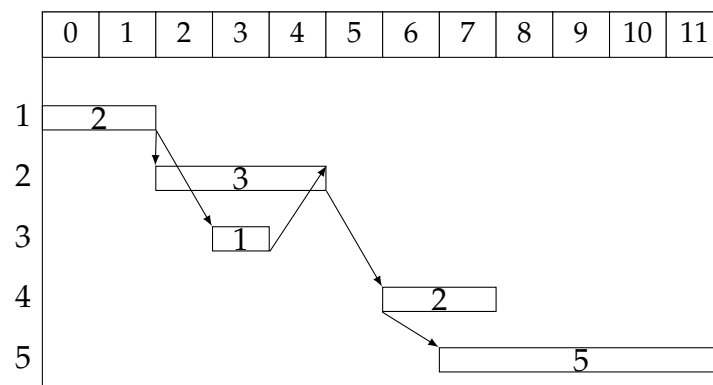
(d) Bestimmen Sie den kritischen Pfad!

Lösungsvorschlag

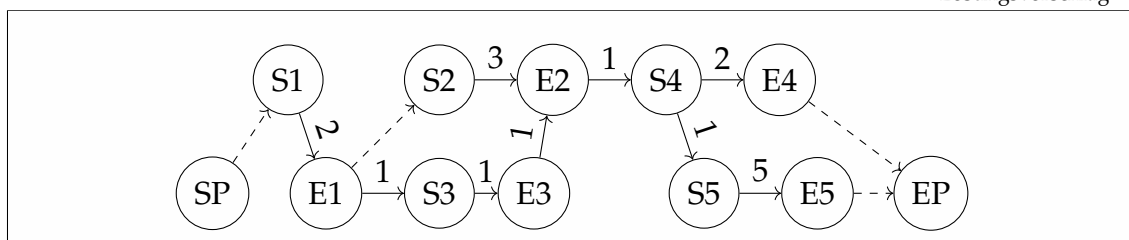
$1 \rightarrow 3 \rightarrow 6 \rightarrow 9 \rightarrow 10$



(e) Konvertieren Sie das Gantt-Diagramm aus Abbildung 3 in ein CPM-Netzwerk!



Lösungsvorschlag

**66116 / 2013 / Herbst / Thema 1 / Teilaufgabe 2 / Aufgabe 3**

- (a) Nennen Sie jeweils einen Vorteil und einen Nachteil für Qualitätssicherung durch „Testing“ bzw. durch „Model Checking“.

Lösungsvorschlag

**Qualitätssicherung durch „Testing“****Vorteil** schnell, Massentests**Nachteil** keine 100% Garantie, dass alles getestet ist**Qualitätssicherung durch „Model Checking“****Vorteil** mathematischer Beweis**Nachteil** teilweise langwierig / nicht möglich

- (b) Definieren Sie den Begriff „Refactoring“.

Lösungsvorschlag

Verbesserung der Code-/Systemstruktur mit dem Ziel einer besseren Wartbarkeit

- Dokumentation, Namen, Kommentare
- keine neuen Funktionalitäten

- bessere Struktur, einheitlich, einfacher

- (c) Begründen Sie, warum bei der Entwicklung nach der Methode des „eXtreme Programming“ langfristig gesehen Refactorings zwingend notwendig werden.

Lösungsvorschlag

Im „eXtreme Programming“ wird das Projekt kontinuierlich aufgebaut, somit ist ein Refactoring, auch aufgrund des Pair-Programmings, auf lange Sicht gesehen notwendig.

- (d) Wie wird in der Praxis während und nach erfolgtem Refactoring sichergestellt, dass keine neuen Defekte eingeführt werden bzw. wurden?

Lösungsvorschlag

Re-testing → erneute Verifikation mit Tests nach Refactoring

- (e) Worin besteht der Unterschied zwischen „White-Box-Testing“ und „Black-Box-Testing“?

Lösungsvorschlag

**White-Box-Testing:** Struktur-Test → Wie funktioniert der Code?

**Black-Box-Testing:** Funktionstest → Das nach außen sichtbare Verhalten wird getestet.

- (f) Nennen Sie vier Qualitätsmerkmale von Software.

Lösungsvorschlag

Änderbarkeit, Wartbarkeit, gute Dokumentation, Effizienz, Funktionalität (→ Korrektheit), Zuverlässigkeit, Portabilität

- (g) Worin besteht der Unterschied zwischen funktionalen und nicht-funktionalen Anforderungen?

Lösungsvorschlag

Funktionale Anforderung: Anforderung an die Funktionalität des Systems, also „Was kann es?“

Nicht-funktionale Anforderung: Design, Programmiersprache, Performanz

- (h) Was verbirgt sich hinter dem Begriff „Continuous Integration“ ?

Lösungsvorschlag

Continuous Integration: Das fertige Modul wird sofort in das bestehende Produkt integriert. Die Integration erfolgt also schrittweise und nicht erst, wenn alle Module fertig sind. Somit können auch neue Funktionalitäten sofort hinzugefügt werden (→ neue Programmversion).

- (i) Nennen Sie sechs Herausstellungsmerkmale des „eXtreme Programming“ Ansatzes.

Lösungsvorschlag

- Werte: Mut, Respekt, Einfachheit, Feedback, Kommunikation
- geringe Bedeutung von formalisiertem Vorgehen, Lösen der Programmieraufgabe im Vordergrund
- fortlaufende Iterationen
- Teamarbeit und Kommunikation (auch mit Kunden)
- Ziel: Software schneller und mit höherer Qualität bereitstellen, höhere Kundenzufriedenheit
- Continuous Integration und Testing, Prototyping
- Risikoanalysen zur Risikominimierung
- YAGNI (You ain't gonna need it) → nur die Features, die gefordert sind, umsetzen; kein Vielleicht braucht man's... "

- (j) Was versteht man unter einem Unit-Test? Begründen Sie, warum es unzureichend ist, wenn eine Test-Suite ausschließlich Unit-Tests enthält.

Lösungsvorschlag

Unter einem Unit-Test versteht man den Test eines einzelnen Software-Moduls oder auch nur einer Methode. Dies ist allein nicht ausreichend, da man so nichts über das Zusammenspiel der Module aussagen kann.

- (k) Nennen Sie jeweils eine Methodik, mit welcher in der Praxis die Prozesse der „Validierung“ und der „Verifikation“ durchgeführt werden.

Lösungsvorschlag

**Methodik der Verifikation:** Testen, wp-Kalkül, Model Checking

**Methodik der Validierung:** Kundentest, Kundengespräch (Spezifiaktion durchsprechen)

- (l) Grenzen Sie die Begriffe „Fault“ und „Failure“ voneinander ab.

Lösungsvorschlag

**Fault:** interner Fehlerzustand, der nach außen nicht sichtbar werden muss, aber kann.

**Failure:** Systemfehler / Fehlerzustand, der nach außen sichtbar wird.

**66116 / 2014 / Herbst / Thema 2 / Teilaufgabe 2 / Aufgabe 2**

Sie sollen das Design für ein einfaches Wahlsystem entwerfen. Das System soll dabei die Verteilung der Stimmen auf die einzelnen Parteien ermöglichen. Zusätzlich soll es verschiedene Darstellungen dieser Daten erlauben: Eine Tabelle, in der die Daten gelesen und auch eingegeben werden können, und ein Diagramm als alternative Darstellung der Informationen. Das System soll mit dem *Model-View-Controller* Muster modelliert werden.

(a) Beschreiben Sie das *Model-View-Controller* Muster:

(i) Beschreiben Sie das Problem, welches das Muster adressiert.

Lösungsvorschlag

Das MVC-Muster wird verwendet, um spätere Änderungen bzw. Erweiterungen zu vereinfachen. Dies unterstützt somit auch die Wiederverwendbarkeit.

(ii) Beschreiben Sie die Aufgaben der Komponenten, die im Muster verwendet werden.

Lösungsvorschlag

Im Modell werden die Daten verwaltet, die View ist für die Darstellung der Daten sowie die Benutzerinteraktion zuständig und der Controller übernimmt die Steuerung zwischen View und Modell.

Das MVC-Muster ist aus den drei Entwurfsmustern Beobachter, Kompositum und Strategie zusammengesetzt.

(b) Modellieren Sie das System unter Anwendung des Musters:

(i) Entwerfen Sie ein UML Klassendiagramm.

(ii) Implementieren Sie eine *setup*-Methode, die das Objektmodell erstellt.

**66116 / 2014 / Herbst / Thema 2 / Teilaufgabe 2 / Aufgabe 3**

Im Folgenden ist ein Algorithmus angegeben, der für eine positive Zahl *until* die Summe aller Zahlen bildet, die kleiner als *until* und Vielfache von 4 oder 6 sind. Für nicht positive Zahlen soll 0 zurückgegeben werden. Der Algorithmus soll also folgender Spezifikation genügen:

$\text{until} > 0 \Rightarrow \text{specialSums}(\text{until}) = \sum \{y \mid 0 < y < \text{until} \wedge (y \% 4 = 0 \vee y \% 6 = 0)\}$

$\text{until} \leq 0 \Rightarrow \text{specialSums}(\text{until}) = 0$

wobei % den Modulo-Operator bezeichnet.

```

public static long specialSums(int until) {
    long sum = 0; // 0
    if (until > 0) { // 1
        for (int i = 1; i <= until; i++) { // 2 // 5
            if (i % 4 == 0 || i % 6 == 0) { // 3
                sum += i; // 4
            }
        }
    }
    return sum; // 6
}

```

Code-Beispiel auf Github ansehen: [src/main/java/org/beschlangaul/examen/examen\\_66116/jahr\\_2014/herbst/SpecialSum.java](https://github.com/src/main/java/org/beschlangaul/examen/examen_66116/jahr_2014/herbst/SpecialSum.java)

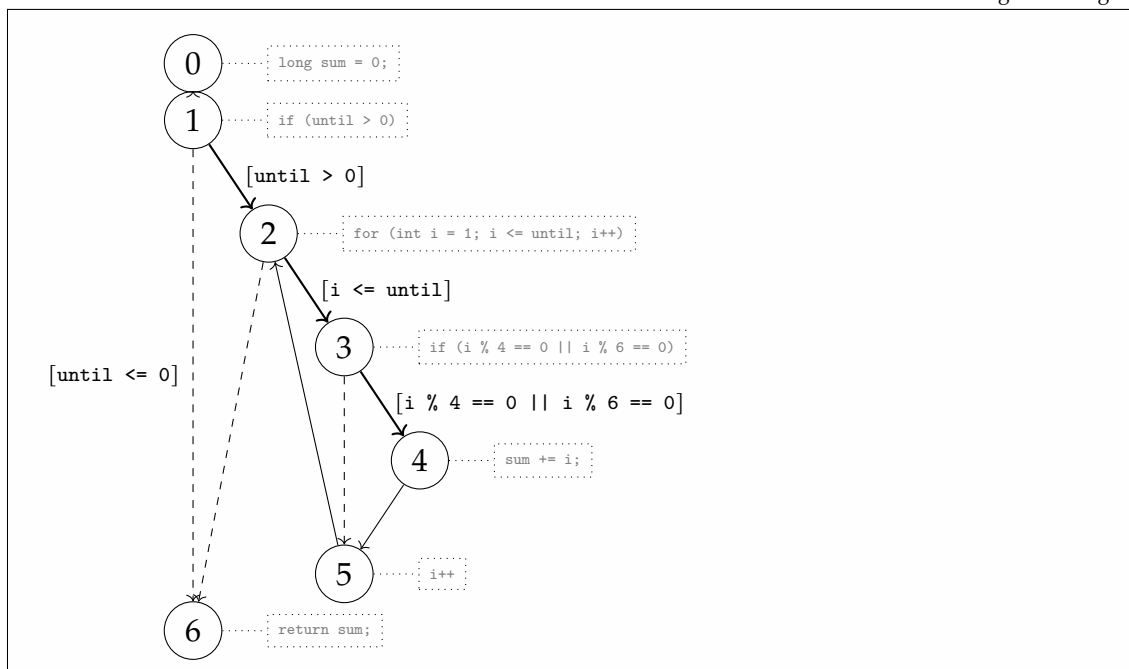
Beachten Sie, dass der Algorithmus nicht der Spezifikation genügt. Der Fehler liegt in der Bedingung der for-Schleife. Der Fehler kann jedoch einfach korrigiert werden indem die Bedingung

$$i \leq \text{until} \text{ in } i < \text{until}$$

geändert würde.

(a) Zeichnen Sie das zum Programm gehörige Ablaufdiagramm.

Lösungsvorschlag



(b) Schreiben Sie einen Testfall, der das Kriterium „100% Anweisungsüberdeckung“ erfüllt, aber den Fehler trotzdem nicht aufdeckt.

Lösungsvorschlag

Der Fehler fällt nur dann auf, wenn until durch 4 oder 6 ohne Rest teilbar ist.  $\text{until} = 0\%4$  oder  $\text{until} = 0\%6$ . Wähle daher den Testfall  $\{(1, 0)\}$ . Alternativ kann für die Eingabe auch 2, 3, 5, 7, 9, 10, 11, 13, ... gewählt werden.

- (c) Schreiben Sie einen Testfall, der das Kriterium „100% Zweigüberdeckung“ erfüllt, aber den Fehler trotzdem nicht aufdeckt.

Lösungsvorschlag

Betrachte den Testfall  $\{(0,0), (5,4)\}$ .

**erste if-Bedingung** Das erste Tupel mit  $\text{until} = 0$  stellt sicher, dass die erste **if**-Bedingung **false** wird.

**Bedingung der for-Schleife** Für die zweite Eingabe  $\text{until} = 5$  werden für  $i$  die Werte 1, 2, 3, 4, 5, 6 angenommen. Wobei für  $i = 6$  die Bedingung der for-Schleife **false** ist.

**Innere if-Bedingung** Für  $i = 1, 2, 3, 5$  wird die innere if-Bedingung jeweils **false**, für  $i = 4$  wird sie **true**.

- (d) Schreiben Sie einen Testfall, der den Fehler aufdeckt. Berechnen Sie Anweisungsüberdeckung und Zweigüberdeckung ihres Testfalls.

Lösungsvorschlag

**Anweisungsüberdeckung** Wähle  $\{(4,0)\}$ . Durch die fehlerhafte Bedingung in der for-Schleife wird der Wert  $i = 4$  akzeptiert. Da alle Anweisungen ausgeführt werden, wird eine Anweisungsüberdeckung mit 100% erreicht.

**Verzweigungsüberdeckung** Da die erste Verzweigung nur zur Hälfte überdeckt wird und die anderen beiden vollständig, gilt für die Verzweigungsüberdeckung:

$$\frac{1+2+2}{2+2+2} = \frac{5}{6}$$

- (e) Es ist nicht immer möglich vollständige Pfadüberdeckung zu erreichen. Geben Sie einen gültigen Pfad des Programmes an, der nicht erreichbar ist. Ein Testfall kann als Menge von Paaren dargestellt werden, wobei jedes Paar  $(I, O)$  die Eingabe  $I$  und die zu dieser erwartete Ausgabe  $O$  darstellt.

Lösungsvorschlag

Ein gültiger Pfad im Kontrollflussgraphen wäre  $\textcircled{0} - \textcircled{1} - \textcircled{2} - \textcircled{3} - \textcircled{4} - \textcircled{5} - \textcircled{2} - \textcircled{6}$ . Der Übergang von  $\textcircled{3}$  auf  $\textcircled{4}$  ist hier aber nicht möglich, da beim ersten Durchlaufen der for-Schleife ( $\textcircled{2}$ )  $i$  immer 1 ist und 1 weder durch 4 noch durch 6 teilbar ist. Somit kann die Bedingung des inneren ifs ( $\textcircled{3}$ ) beim ersten Durchlauf nie *wahr* sein, womit immer der Übergang  $\textcircled{3} - \textcircled{5}$  zu Beginn genommen werden muss. Alle Pfade, die zu Beginn  $\textcircled{3} - \textcircled{4}$  enthalten, sind somit nicht überdeckbar.

## 66116 / 2015 / Frühjahr / Thema 1 / Teilaufgabe 2 / Aufgabe 2

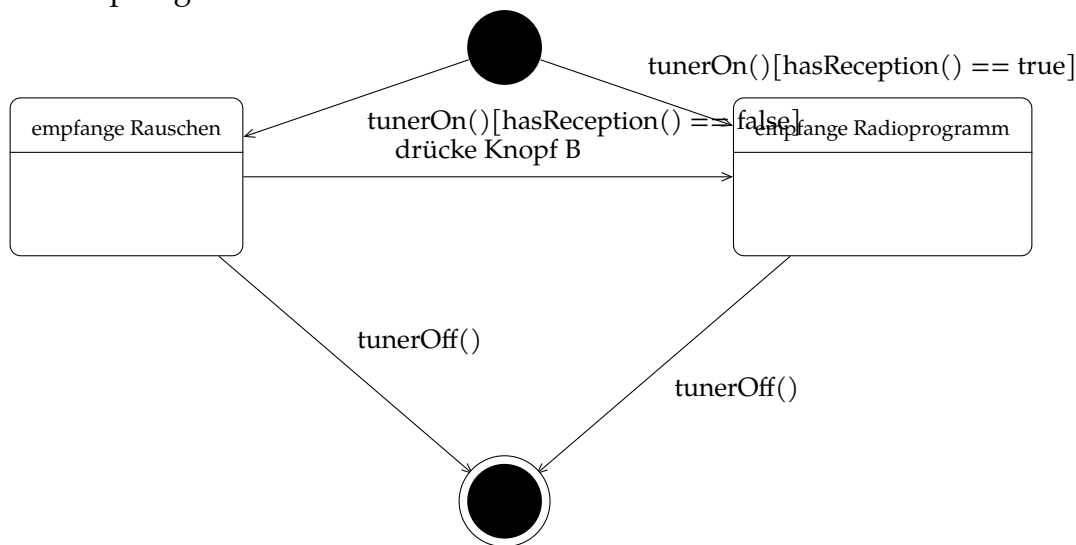
Erstellen Sie ein UML-Zustandsdiagramm für einen Radiotuner. Mit dem Radiotuner können Sie ein Radioprogramm auf der Frequenz  $f$  empfangen. Beim Einschalten wird

f auf 87,5 MHz gesetzt und der Tuner empfängt. Sie können nun die Frequenz in 0,5 MHz Schritten erhöhen oder senken. Bitte beachten Sie, dass das Frequenzband für den Radioempfang von 87,5 MHz bis 108 MHz reicht. Sobald f 87,5 MHz unter- bzw. 108 MHz überschreitet, soll f auf 108 MHz bzw. 87,5 MHz gesetzt werden. Weiterhin kann ein Suchmodus gestartet werden, der automatisch die Frequenz erhöht, bis ein Sender empfangen wird. Wird während des Suchmodus die Frequenz verändert oder erneut Suchen ausgeführt, dann wird die Suche beendet (und, je nach Knopf, ggf. noch die Frequenz um 0,5 MHz verändert).

Die Klasse RadioTuner besitzt folgende Methoden:

- tunerOn()
- tunerOff()
- increaseFrequency()
- decreaseFrequency()
- seek()
- hasReception()

Hinweis: Die Hilfsmethode hasReception() liefert true zurück, genau dann wenn ein Sender empfangen wird.



## 66116 / 2015 / Frühjahr / Thema 1 / Teilaufgabe 2 / Aufgabe 3

In dieser Aufgabe implementieren Sie ein konzeptionelles Datenmodell für eine Firma, die Personendaten von Angestellten und Kunden verwalten möchte. Gegeben seien dazu folgende Aussagen:

- Eine *Person* hat einen *Namen* und ein *Geschlecht* (männlich oder weiblich).
- Ein *Angestellter* ist eine *Person*, zu der zusätzlich das monatliche *Gehalt* gespeichert wird.



- Ein *Kunde* ist eine *Person*, zu der zusätzlich eine *Kundennummer* hinterlegt wird.
- (a) Geben Sie in einer objektorientierten Programmiersprache Ihrer Wahl (geben Sie diese an) eine Implementierung des aus den obigen Aussagen resultierenden konzeptionellen Datenmodells in Form von **Klassen** und **Interfaces** an. Gehen Sie dabei wie folgt vor:
- Schreiben Sie ein Interface `Person` sowie zwei davon erbbende Interfaces `Angestellter` und `Kunde`. Die Interfaces sollen jeweils lesende Zugriffsmethoden (Getter) die entsprechenden Attribute (Name, Geschlecht, Gehalt, Kundennummer) deklarieren.

Lösungsvorschlag

```
public interface Person {  
    String getName();  
  
    char getGeschlecht();  
}
```

Code-Beispiel auf Github ansehen: [src/main/java/org/bschlangaul/examen/examen\\_66116/jahr\\_2015/fruehjahr/Person.java](https://github.com/bschlangaul/examen/examen_66116/jahr_2015/fruehjahr/Person.java)

```
public interface Angestellter extends Person {  
    int getGehalt();  
}
```

Code-Beispiel auf Github ansehen: [src/main/java/org/bschlangaul/examen/examen\\_66116/jahr\\_2015/fruehjahr/Angestellter.java](https://github.com/bschlangaul/examen/examen_66116/jahr_2015/fruehjahr/Angestellter.java)

```
public interface Kunde extends Person {  
    int getKundennummer();  
}
```

Code-Beispiel auf Github ansehen: [src/main/java/org/bschlangaul/examen/examen\\_66116/jahr\\_2015/fruehjahr/Kunde.java](https://github.com/bschlangaul/examen/examen_66116/jahr_2015/fruehjahr/Kunde.java)

- Schreiben Sie eine abstrakte Klasse `PersonImpl`, die das Interface `Person` implementiert. Für jedes Attribut soll ein Objektfeld angelegt werden. Außerdem soll ein Konstruktor definiert werden, der alle Objektfelder initialisiert.

Lösungsvorschlag

```
public abstract class PersonImpl implements Person {  
    protected String name;  
    protected char geschlecht;  
  
    public PersonImpl(String name, char geschlecht) {  
        this.name = name;  
        this.geschlecht = geschlecht;  
    }  
  
    public String getName() {  
        return name;  
    }  
  
    public char getGeschlecht() {  
        return geschlecht;  
    }  
}
```

}

Code-Beispiel auf Github ansehen: [src/main/java/org/bschlangaul/examen/examen\\_66116/jahr\\_2015/fruehjahr/PersonImpl.java](https://github.com/bschlangaul/examen/examen_66116/jahr_2015/fruehjahr/PersonImpl.java)

- Schreiben Sie zwei Klassen `AngestellterImpl` und `KundeImpl`, die von `PersonImpl` erben und die jeweils dazugehörigen Interfaces implementieren. Es sollen wiederum Konstruktoren definiert werden, die alle Objektfelder initialisieren und dabei auf den Konstruktor der Basisklasse `PersonImpl` Bezug nehmen.

Lösungsvorschlag

```
public class AngestellterImpl extends PersonImpl implements Angestellter
{
    protected int gehalt;

    public AngestellterImpl(String name, char geschlecht, int gehalt) {
        super(name, geschlecht);
        this.gehalt = gehalt;
    }

    public int getGehalt() {
        return gehalt;
    }
}
```

Code-Beispiel auf Github ansehen: [src/main/java/org/bschlangaul/examen/examen\\_66116/jahr\\_2015/fruehjahr/AngestellterImpl.java](https://github.com/bschlangaul/examen/examen_66116/jahr_2015/fruehjahr/AngestellterImpl.java)

```
public class KundeImpl extends PersonImpl implements Kunde {
    protected int kundennummer;

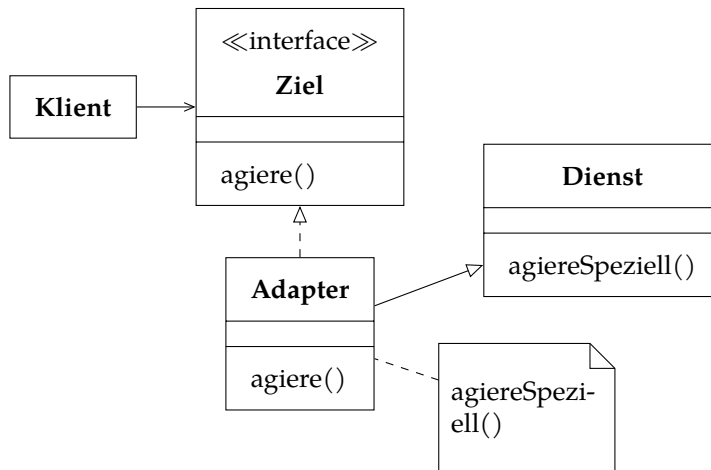
    public KundeImpl(String name, char geschlecht, int kundennummer) {
        super(name, geschlecht);
        this.kundennummer = kundennummer;
    }

    public int getKundennummer() {
        return kundennummer;
    }
}
```

Code-Beispiel auf Github ansehen: [src/main/java/org/bschlangaul/examen/examen\\_66116/jahr\\_2015/fruehjahr/KundeImpl.java](https://github.com/bschlangaul/examen/examen_66116/jahr_2015/fruehjahr/KundeImpl.java)

- (b) Verwenden Sie das Entwurfsmuster **Adapter**, um zu ermöglichen, dass vorhandene Angestellte die Rolle eines Kunden einnehmen können. Der Adapter soll eine zusätzliche Klasse sein, die das Kunden-Interface implementiert. Wenn möglich, sollen Methodenaufrufe an den adaptierten Angestellten delegiert werden. Möglicherweise müssen Sie neue Objektfelder einführen.

### Exkurs: Entwurfsmuster „Adapter“



**Ziel (Target)** Das Ziel definiert die Schnittstelle, die der Klient nutzen kann.

**Klient (Client)** Der Klient nutzt Dienste über inkompatible Schnittstellen und greift dabei auf adaptierte Schnittstellen zurück.

**Dienst (Adaptee)** Der Dienst bietet wiederzuverwendende Dienstleistungen mit fest definierter Schnittstelle an.

**Adapter** Der Adapter adaptiert die Schnittstelle des Dienstes auf die Schnittstelle zum Klienten.

```

/**
 * GoF: Adaptee
 */
public class Dienst {

    public void agiereSpeziell() {
        System.out.println("Agiere speziell!");
    }
}

```

Code-Beispiel auf Github ansehen: [src/main/java/org/bschlangaul/entwurfsmuster/adapter/allgemein/Dienst.java](https://github.com/bschlangaul/entwurfsmuster/tree/master/adapter/allgemein/Dienst.java)

```

public interface Ziel {
    public void agiere();
}

```

Code-Beispiel auf Github ansehen: [src/main/java/org/bschlangaul/entwurfsmuster/adapter/allgemein/Ziel.java](https://github.com/bschlangaul/entwurfsmuster/tree/master/adapter/allgemein/Ziel.java)

```

public class Adapter extends Dienst implements Ziel {

    @Override
    public void agiere() {
        System.out.print("agiere: ");
        agiereSpeziell();
    }
}

```

Code-Beispiel auf Github ansehen: [src/main/java/org/bschlangaul/entwurfsmuster/adapter/allgemein/Adapter.java](https://github.com/bschlangaul/entwurfsmuster/tree/master/adapter/allgemein/Adapter.java)

```

public class Klient {

    public static void main(String[] args) {
        new Adapter().agiere();
        // agiere: Agiere speziell!
    }
}

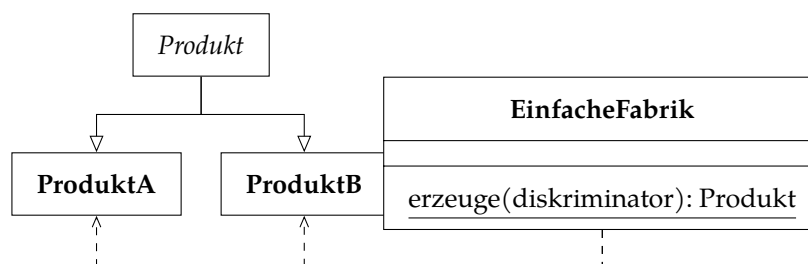
```

Code-Beispiel auf Github ansehen: [src/main/java/org/bschlangaul/entwurfsmuster/adapter/allgemein/Klient.java](https://github.com/bschlangaul/entwurfsmuster/adapter/allgemein/Klient.java)

- (c) Verwenden Sie das Entwurfsmuster **Simple Factory**, um die Erzeugung von Angestellten, Kunden, sowie Adapter-Instanzen aus Aufgabe (b) zu vereinheitlichen. Die entsprechende Erzeugungs-Methode soll neben einem Typ-Diskriminator (z. B. einem Aufzählungstypen oder mehreren booleschen Werten) alle Parameter übergeben bekommen, die für den Konstruktor irgendeiner Implementierungsklasse des Interface **Person** notwendig sind.

Hinweis: Um eine Adapter-Instanz zu erzeugen, müssen Sie möglicherweise zwei Konstruktoren aufrufen.

#### Exkurs: Entwurfsmuster „Einfache Fabrik“



**EinfacheFabrik** Eine Klasse mit einer Erzeugungsmethode, die über eine größere Bedingung verschiedene Objekt instanziert.

**Produkt** Eine abstrakte Klasse, die von den konkreten Produkten geerbt wird.

**KonkretesProdukt** Ein konkretes Produkt, das von der einfachen Fabrik erzeugt wird.

## 66116 / 2015 / Frühjahr / Thema 2 / Teilaufgabe 2 / Aufgabe 2

Leider ist das Klassendiagramm der folgenden Klassen verloren gegangen. Führen Sie ein Reverse Engineering durch und erstellen Sie aus dem Quellcode ein vollständiges UML-Klassendiagramm inklusive aller Klassen, Schnittstellen, Attribute, Methoden, Konstruktoren, Sichtbarkeiten, Assoziationen, Rollennamen, Multiplizitäten, Navigationspfeilen und evtl. Stereotypen. Der Quellcode innerhalb von Methoden und Konstruktoren soll nicht übertragen werden, wohl aber die Methodensignaturen. Assoziationsnamen und deren Leserichtung lassen sich aus dem Quellcode nur schwer erraten und sollen deshalb ebenfalls weggelassen werden.

```

public abstract class Display implements PixelPainter {
    protected HardwareMatrix hardwareMatrix;
}

```

```
protected int lastPaintedX;
protected int lastPaintedY;

public Display(HardwareMatrix hardwareMatrix) {
    this.hardwareMatrix = hardwareMatrix;
}

public int getWidth() {
    return hardwareMatrix.getWidth() / getWidthFactor();
}

public int getHeight() {
    return hardwareMatrix.getHeight() / getHeightFactor();
}

public void clear() {
    // some longer code
}

protected abstract int getWidthFactor();

protected abstract int getHeightFactor();
}
```

Code-Beispiel auf Github ansehen: [src/main/java/org/bschlangaul/examen/examen\\_66116/jahr\\_2015/fruehjahr/reverse/Display.java](#)

```
import java.awt.Color;

public interface PixelPainter {
    void set(int x, int y, Color color);

    int getHeight();

    int getWidth();
}
```

Code-Beispiel auf Github ansehen: [src/main/java/org/bschlangaul/examen/examen\\_66116/jahr\\_2015/fruehjahr/reverse/PixelPainter.java](#)

```
public interface HardwareMatrix {
    void set(int x, int y, int v);

    int getWidth();

    int getHeight();
}
```

Code-Beispiel auf Github ansehen: [src/main/java/org/bschlangaul/examen/examen\\_66116/jahr\\_2015/fruehjahr/reverse/HardwareMatrix.java](#)

```
import java.awt.Color;

@SuppressWarnings({ "unused" })
public class DisplayUnion extends RGBDisplay {
    public static final int MAX_DISPLAY_COUNT = 50;

    private int currentDisplayCount;
```

```
private Display[] displays;

public DisplayUnion(Display[] displays) {
    super(null);
}

public int getDisplayCount() {
    return 0;
}

protected int getWidthFactor() {
    return 1;
}

protected int getHeightFactor() {
    return 1;
}

public void set(int x, int y, Color color) {
}
}
```

Code-Beispiel auf Github ansehen: [src/main/java/org/bschlangaul/examen/examen\\_66116/jahr\\_2015/fruehjahr/reverse/DisplayUnion.java](https://github.com/bschlangaul/examen/examen_66116/jahr_2015/fruehjahr/reverse/DisplayUnion.java)

```
import java.awt.Color;

public class RGBDisplay extends Display {
    public RGBDisplay(HardwareMatrix matrix) {
        super(matrix);
    }

    public void set(int x, int y, Color color) {
    }

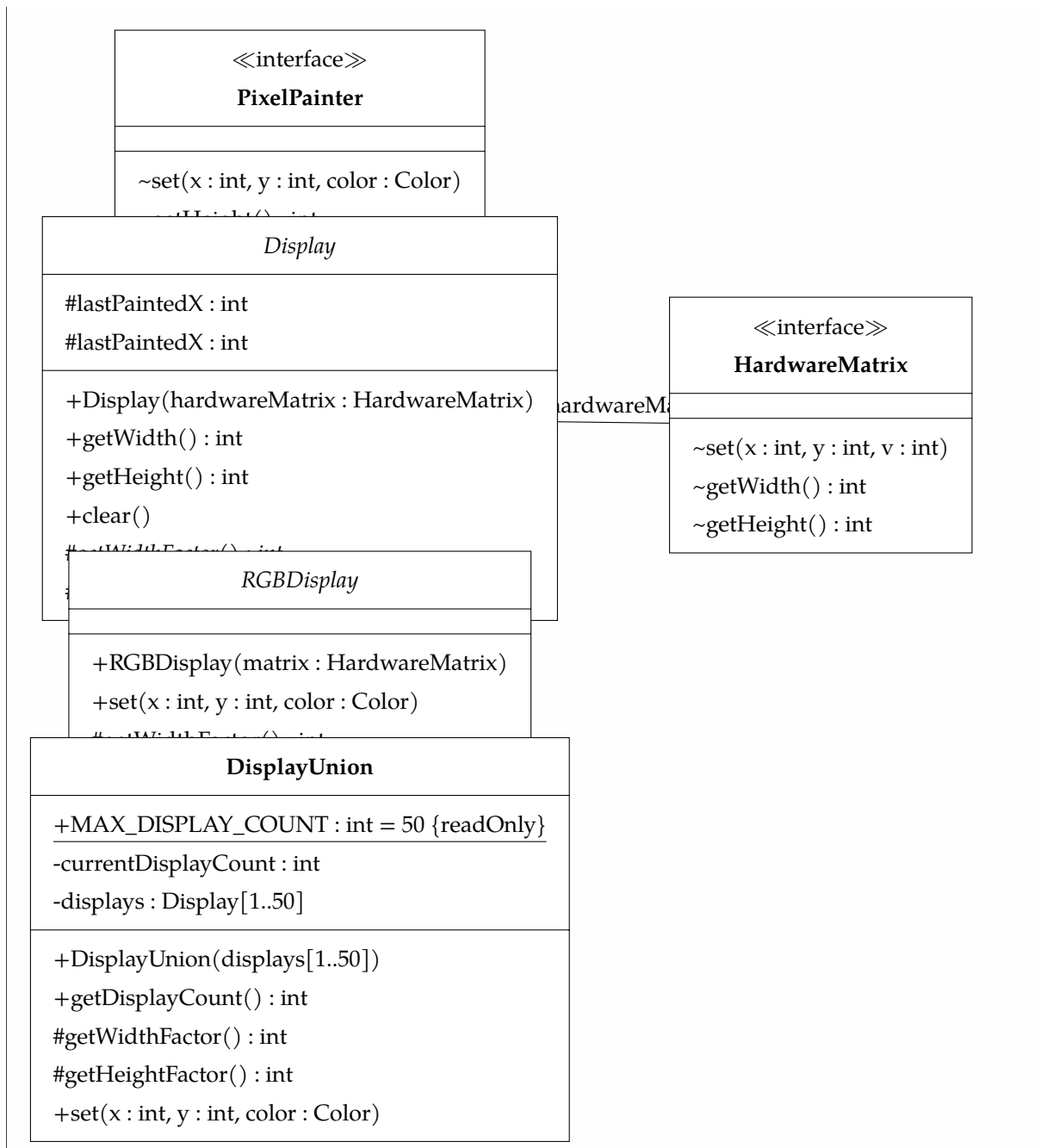
    protected int getWidthFactor() {
        return 3;
    }

    protected int getHeightFactor() {
        return 1;
    }
}
```

Code-Beispiel auf Github ansehen: [src/main/java/org/bschlangaul/examen/examen\\_66116/jahr\\_2015/fruehjahr/reverse/RGBDisplay.java](https://github.com/bschlangaul/examen/examen_66116/jahr_2015/fruehjahr/reverse/RGBDisplay.java)

Lösungsvorschlag

---

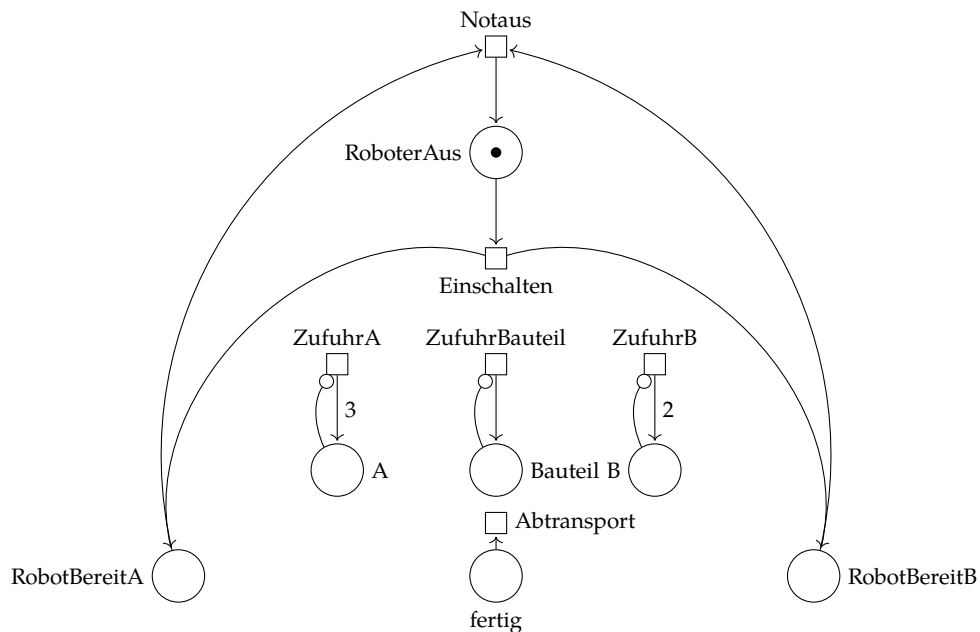


## 66116 / 2015 / Frühjahr / Thema 2 / Teilaufgabe 2 / Aufgabe 3

Das folgende Grundgerüst stammt aus dem Petri-Netz-Modell einer Automatisierungsanlage mit zwei Robotern, das Sie auf Ihr Blatt übernehmen und geeignet um weitere Plätze (Stellen), Transitionen, Kapazitäten, Gewichte und Markierungen so ergänzen sollen, dass die darunter angegebenen Anforderungen eingehalten werden:

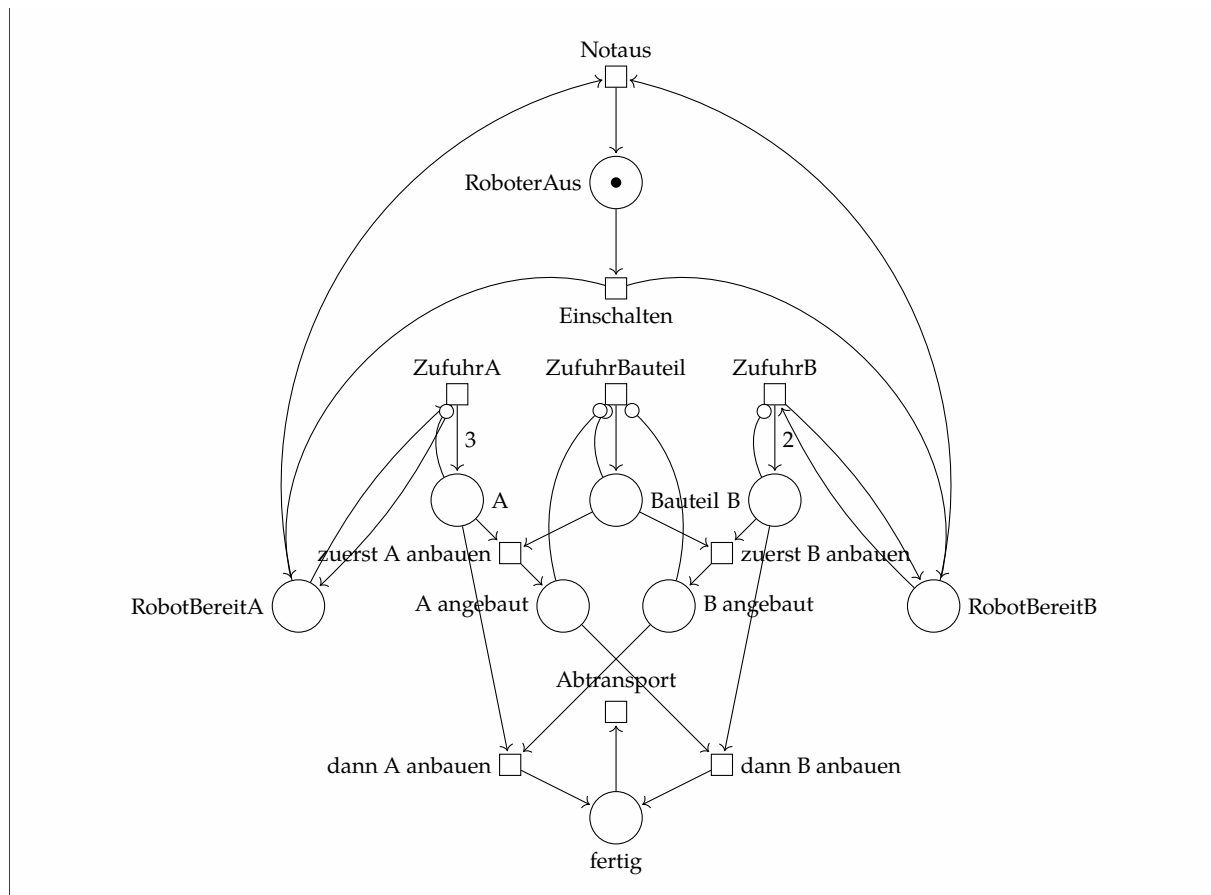
In der Anlage arbeiten zwei Roboter A und B, die über einen Schalter „Einschalten“ aktiviert und *jederzeit* über einen „Notaus“-Schalter deaktiviert werden können müssen. Aufgabe der Roboter ist es, jeweils abwechselnd an Bauteilen zu arbeiten, die einzeln über ein Förderband „ZufuhrBauteil“ ins System eingefahren und nach der Fer-

tigstellung mittels „Abtransport“ aus dem System abgeführt werden. Roboter A bringt genau 3 Anbauten vom Typ „A“ an jedes Bauteil an und Roboter B macht das entsprechend mit genau 2 Anbauten vom Typ „B“. Die Anbauten werden jeweils passend über „ZufuhrA“ auf „A“ bzw. mittels „ZufuhrB“ auf „B“ bereitgestellt. Die beiden Roboter dürfen *niemals* gleichzeitig an einem Bauteil arbeiten — die Reihenfolge, in der sie darauf zugreifen, ist jedoch beliebig und darf von Bauteil zu Bauteil frei variieren. Sobald einer der Roboter mit der Arbeit an einem neuen Bauteil begonnen hat, darf kein weiteres Bauteil angenommen werden, ehe der jeweils andere Roboter nicht ebenfalls mit seiner Arbeit am gleichen Bauteil fertig geworden ist (und das fertige Bauteil „auf den Platz ‚fertig‘ legt“). Aus Platzgründen darf höchstens ein Bauteil auf dem Ausgangsplatz „fertig“ abgestellt werden, ehe es abtransportiert wird.



Lösungsvorschlag





## 66116 / 2015 / Herbst / Thema 2 / Teilaufgabe 2 / Aufgabe 3

Gegeben Sei folgendes Programm:

```

long doubleFac (long n) {
    /* P */ long df = 1;
    for (long x = n; x > 1; x -= 2) {
        df *= x;
    } /* Q */
    return df;
}

```

sowie die Vorbedingung  $P \equiv n \geq 0$  und Nachbedingung  $Q \equiv (df = n!!)$  wobei gilt

$$n!! := \begin{cases} 2^k \cdot k! & n \text{ gerade, } k := \frac{n}{2} \\ \frac{(2k)!}{2^k \cdot k!} & n \text{ ungerade, } k := \frac{n+1}{2} \end{cases}$$

### Exkurs: Fakultät

Die Fakultät ist eine Funktion, die einer natürlichen Zahl das Produkt aller natürlichen Zahlen (ohne Null) kleiner und gleich dieser Zahl zuordnet. Für alle natürlichen Zahlen  $n$  ist

$$n! = 1 \cdot 2 \cdot 3 \cdot \dots \cdot n = \prod_{k=1}^n k$$

**Exkurs: Doppelfakultät**

Die seltener verwendete „Doppelfakultät“ oder „doppelte Fakultät“ ist für gerade  $n$  das Produkt aller geraden Zahlen kleiner gleich  $n$ . Für ungerade  $n$  ist es das Produkt aller ungeraden Zahlen kleiner gleich  $n$ . Sie ist definiert als:

$$n!! = \begin{cases} n \cdot (n-2) \cdot (n-4) \cdot \dots \cdot 2 & \text{für } n \text{ gerade und } n > 0, \\ n \cdot (n-2) \cdot (n-4) \cdot \dots \cdot 1 & \text{für } n \text{ ungerade und } n > 0, \\ 1 & \text{für } n \in \{-1, 0\} \end{cases}$$

Häufig werden anstelle der Doppelfakultät Ausdrücke mit der gewöhnlichen Fakultät verwendet.

Es gilt  $(2k)!! = 2^k k!$  und  $(2k-1)!! = \frac{(2k)!}{2^k k!}$

Zur Vereinfachung nehmen Sie im Folgenden an, dass die verwendeten Datentypen unbeschränkt sind und daher keine Überläufe auftreten können.

(a) Welche der folgenden Bedingungen ist eine zum Beweisen der Korrektheit der Methode mittels wp-Kalkül (Floyd-Hoare-Kalkül) sinnvolle Schleifeninvariante?

- (i)  $df = n!! - x!! \wedge x \geq 1$
- (ii)  $df = (n - x)!! \wedge x \geq 1$
- (iii)  $df \cdot x!! = n!! \wedge x \geq 0$
- (iv)  $(df + x)!! = n!! \wedge x \geq 0$

Lösungsvorschlag

Zunächst wird der Code in einen äquivalenten Code mit while-Schleife umgewandelt:

```
long doubleFac (long n) {
    /* P */ long df = 1;
    long x = n ;
    while (x > 1) {
        df = df * x;
        x = x - 2;
    } /* Q */
    return df;
}
```

- (i)  $df = n!! - x!! \wedge x \geq 1$
- (ii)  $df = (n - x)!! \wedge x \geq 1$

Die ersten beiden Bedingungen sind unmöglich, da z. B. für  $n = 2$  nach der Schleife  $x = 0$  gilt und daher  $x \geq 1$  verletzt wäre.

$$(iii) \text{ df} \cdot x!! = n!! \wedge x \geq 0$$

Nach dem Ausschlussprinzip ist es daher die dritte Bedingung:  $I \equiv (\text{df} + x)!! = n!! \wedge x \geq 0$ .

$$(iv) (\text{df} + x)!! = n!! \wedge x \geq 0$$

Die letzte kann es auch nicht sein, da vor der Schleife  $\text{df} = 1$  und  $x = n$  gilt,  $\delta(\text{df} + x)!! = (1 + n)!!$ . Jedoch ist offenbar  $(1 + n)!! \neq n!!$ .

$\Rightarrow$  Die Schleifeninvariante lautet:  $\text{df} \cdot x!! = n!! \wedge x \geq 0$

- (b) Zeigen Sie formal mittels wp-Kalkül, dass die von Ihnen gewählte Bedingung unmittelbar vor Beginn der Schleife gilt, wenn zu Beginn der Methode die Anfangsbedingung  $P$  gilt.

Lösungsvorschlag

Zu zeigen  $P \Rightarrow \text{wp}(\text{"Code vor der Schleife"}, I)$

$$\begin{aligned} \text{wp}(\text{"Code vor der Schleife"}, I) &\equiv \text{wp}(\text{"df} = 1; x = n; \text{"}, (\text{df} \cdot x)!! = n!! \wedge x \geq 0) \\ &\equiv \text{wp}(\text{"df} = 1; \text{"}, (\text{df} \cdot n)!! = n!! \wedge n \geq 0) \\ &\equiv \text{wp}(\text{"}, (1 \cdot n)!! = n!! \wedge n \geq 0) \\ &\equiv n!! = n!! \wedge n \geq 0 \\ &\equiv n \geq 0 \\ &\equiv P \end{aligned}$$

Insbesondere folgt damit die Behauptung.

- (c) Zeigen Sie formal mittels wp-Kalkül, dass die von Ihnen gewählte Bedingung tatsächlich eine Invariante der Schleife ist.

Lösungsvorschlag

zu zeigen:  $I \wedge \text{Schleifenbedingung} \Rightarrow \text{wp}(\text{"Code in der Schleife"}, I)$

Bevor wir dies beweisen, zeigen wir erst  $x \cdot (x - 2)!! = x!!$ .

- Fall  $x$  ist gerade ( $n!! = 2^k \cdot k!$  für  $k := \frac{n}{2}$ ):

$$x \cdot (x - 2)!! = x \cdot 2^{\frac{x-2}{2}} \cdot (\frac{x-2}{2})! = x \cdot \frac{1}{2} \cdot 2^{\frac{x}{2}} \cdot (\frac{x}{2} - 1)! = 2^{\frac{x}{2}} \cdot (\frac{x}{2})! = x!!$$

Nebenrechnung (Division mit gleicher Basis:  $x^{a-b} = \frac{x^a}{x^b}$ ):

$$2^{\frac{x-2}{2}} = 2^{(\frac{x}{2}-\frac{2}{2})} = \frac{2^{\frac{x}{2}}}{2^{\frac{2}{2}}} = \frac{2^{\frac{x}{2}}}{2^1} = \frac{2^{\frac{x}{2}}}{2} = \frac{1}{2} \cdot 2^{\frac{x}{2}}$$

Nebenrechnung ( $n! = (n - 1)! \cdot n$ ):

$$x \cdot \frac{1}{2} \cdot (\frac{x}{2} - 1)! = \frac{x}{2} \cdot (\frac{x}{2} - 1)! = \frac{x}{2}!$$

- Fall  $x$  ist ungerade:

Dies benutzen wir nun, um den eigentlichen Beweis zu führen:

$$\begin{aligned}
 \text{wp}(\text{"Code vor der Schleife"}, I) &\equiv \text{wp}(\text{"df = df * x; x = x - 2;"}, (\text{df} \cdot x)!! = n!! \wedge x \geq 0) \\
 &\equiv \text{wp}(\text{"df = df * x;"}, (\text{df} \cdot (x - 2))!! = n!! \wedge x - 2 \geq 0) \\
 &\equiv \text{wp}(\text{"", } (\text{df} \cdot x \cdot (x - 2))!! = n!! \wedge x - 2 \geq 0) \\
 &\equiv (\text{df} \cdot x)!! = n!! \wedge x \geq 2 \\
 &\equiv (\text{df} \cdot x)!! = n!! \wedge x > 1 \\
 &\equiv I \wedge x > 1 \\
 &\equiv I \wedge \text{Schleifenbedingung}
 \end{aligned}$$

- (d) Zeigen Sie formal mittels wp-Kalkül, dass am Ende der Methode die Nachbedingung  $Q$  erfüllt wird.

Lösungsvorschlag

$$\text{z.z. } I \wedge \neg \text{Schleifenbedingung} \Rightarrow \text{wp}(\text{"Code nach der Schleife"}, Q)$$

Wir vereinfachen den Ausdruck  $I \wedge \neg \text{Schleifenbedingung}$ :

$$I \wedge \neg \text{Schleifenbedingung} \equiv I \wedge (x \leq 1) \equiv I \wedge ((x = 0) \vee (x = 1)) \equiv (I \wedge (x = 0)) \vee (I \wedge (x = 1)) \equiv (\text{df} \cdot 1 = n!!) \vee (\text{df} \cdot 1 = n!!) \equiv \text{df} = n!!$$

Damit gilt:

$$\text{wp}(\text{"Code nach der Schleife"}, Q) \equiv \text{wp}(\text{"", } \text{df} = n!!) \equiv \text{df} = n!! \equiv I \wedge \neg \text{Schleifenbedingung}$$

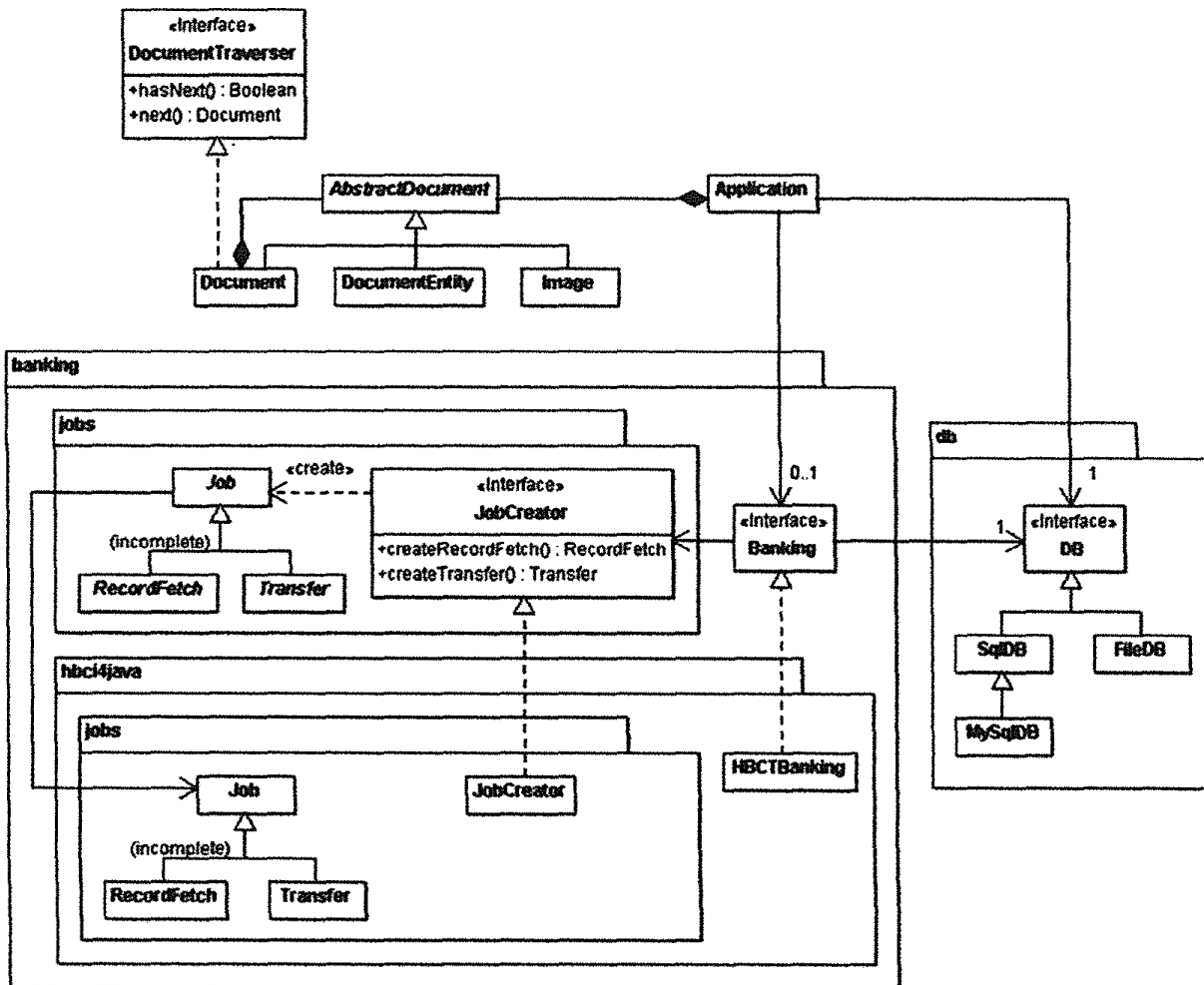
- (e) Beweisen Sie, dass die Methode immer terminiert. Geben Sie dazu eine Terminierungsfunktion an und begründen Sie kurz ihre Wahl.

Lösungsvorschlag

Sei  $T(x) := x$ .  $T$  ist offenbar ganzzahlig. Da  $x$  in jedem Schleifendurchlauf um 2 verringert wird, ist  $T$  streng monoton fallend. Aus der Schleifeninvariante folgt  $x \geq 0$  und daher ist  $x$  auch nach unten beschränkt. Damit folgt  $I \Rightarrow T \geq 0$  und  $T$  ist eine gültige Terminierungsfunktion.

## 66116 / 2016 / Frühjahr / Thema 1 / Teilaufgabe 2 / Aufgabe 2

Klassendiagramm  
 Abstrakte Fabrik (Abstract  
 Factory)  
 Wiederholer (Iterator)  
 Adapter  
 Kompositum (Composite)



- (a) Kennzeichnen Sie im folgenden Klassendiagramm die Entwurfsmuster „Abstrakte Fabrik“, „Iterator“, „Adapter“ und „Kompositum“. Geben Sie die jeweils beteiligten Klassen und deren Zuständigkeit im entsprechenden Muster an.

Lösungsvorschlag

**Iterator**

**DocumentTraverser (interface)** Schnittstelle zur Traversierung und zum Zugriff auf Dokumente

**Document** implementiert die Schnittstelle

**Kompositum**

**AbstractDocument** abstrakte Basisklasse, die gemeinsames Verhalten der beteiligten Klassen definiert

**Document** enthält wiederum weitere Dokumente bzw. DocumentEntities und Images

**DocumentEntity, Image** primitive Unterklassen, besitzen keine Kindobjekte

**Adapter (Objektadapter)**

**Banking (interface)** vom Client (hier Application) verwendete Schnittstelle

**HBCTBanking** passt Schnittstelle der unpassenden Klasse an Zielschnittstelle (Banking) an

**DB (interface)** anzupassende Schnittstelle

**abstrakte Fabrik**

**JobCreator (interface)** abstrakte Fabrik

**Job (abstrakt) mit Unterklassen RecordFetch und Transfer** abstraktes Produkt

**Job (konkret) mit Unterklassen** konkretes Produkt

- (b) (i) Beschreiben Sie die Funktionsweise der folgenden Entwurfsmuster und geben Sie ein passendes UML-Diagramm an.
- Dekorierer
  - Klassenadapter
  - Objektadapter
- (ii) Erklären Sie mit maximal zwei Sätzen den Unterschied zwischen Klassenadapter und Objektadapter.
- (c) Implementieren Sie einen Stapel in der Programmiersprache Java. Nutzen Sie dazu ein Array mit fester Größe. Auf eine Überlaufprüfung darf verzichtet werden. Implementieren Sie in der Klasse das Iterator Entwurfsmuster, um auf die Inhalte zuzugreifen, sowie eine Funktion zum Hinzufügen von Elementen. Als Typ für den Stapel kann zur Vereinfachung ein Integertyp verwendet werden.

**66116 / 2016 / Frühjahr / Thema 2 / Teilaufgabe 2 / Aufgabe 4**

- (a) Erklären Sie in maximal zwei Sätzen den Unterschied zwischen Knoten- und Kantennetzwerken im Kontext des Projektmanagements.
- (b) Gegeben ist die folgende Tabelle zur Grobplanung eines hypothetischen Softwareprojekts:

Aktivität	Minimale Dauer	Einschränkungen
Anforderungsanalyse	2 Monate	Endet frühestens einen Monat nach dem Start der Entwurfsphase.
Entwurf	4 Monate	Startet frühestens zwei Monate nach dem Start der Anforderungsanalyse.
Implementierung	5 Monate	Endet frühestens drei Monate nach dem Ende der Entwurfsphase. Darf erst starten, nachdem die Anforderungsanalyse abgeschlossen ist.

Geben Sie ein CPM-Netzwerk an, das die Aktivitäten und Abhängigkeit des obigen Projektplans beschreibt. Gehen Sie von der Zeiteinheit „Monate“ aus. Das Projekt hat einen Start- und einen Endknoten.

Jede Aktivität wird auf einen Start- und einen Endknoten abgebildet. Die Dauer der Aktivitäten sowie Abhängigkeiten sollen durch Kanten dargestellt werden. Der Start jeder Aktivität hängt vom Projektstart ab, das Projektende hängt vom Ende aller Aktivitäten ab. Modellieren Sie diese Abhängigkeiten durch Pseudoaktivitäten mit Dauer null.

- (c) Berechnen Sie für jedes Ereignis (für jeden Knoten) die früheste Zeit sowie die späteste Zeit. Beachten Sie, dass die Berechnungsreihenfolge einer topologischen Sortierung des Netzwerks entsprechen sollte.
- (d) Geben Sie einen kritischen Pfad durch das CPM-Netzwerk an. Möglicherweise sind hierfür weitere Vorberechnungen vonnöten. Welche Aktivität sollte sich demnach auf keinen Fall verzögern?
- (e) Geben Sie ein Gantt-Diagramm an, das den Projektplan visualisiert. Gehen Sie davon aus, dass jede Aktivität zur frühesten Zeit ihres Startknotens beginnt und zur spätesten Zeit ihres Endknotens endet (s. jeweils Teilaufgabe (c)). Geben Sie die minimale Dauer jeder Aktivität, sowie die Pufferzeit (in Klammern) an. Beispiel: 4 (+2). Notieren Sie alle Einschränkungen mit Hilfe geeigneter Abhängigkeitsbeziehungen. Geben Sie eine absolute Zeitskala in Monaten an.
- (f) Nennen Sie zwei weitere Aktivitäten, die in der obigen Tabelle fehlen, jedoch typischerweise in Softwareentwicklungs-Prozessmodellen wie etwa dem Wasserfallmodell vorkommen.

## 66116 / 2016 / Herbst / Thema 1 / Teilaufgabe 2 / Aufgabe 1

- (a) Welche Kriterien werden bei der Zielbestimmung im Pflichtenheft unterschieden?

Lösungsvorschlag

Im Rahmen der Zielbestimmung werden die Ziele des Produktes in einer Art Prioritätenliste exakt eingegrenzt, die in drei weitere Kategorien gegliedert ist, die als **Muss-**, **Wunsch-** und **Abgrenzungskriterien** bezeichnet werden.

**Musskriterien** Zu den Musskriterien zählen die Produktfunktionen, die für ein Funktionieren des Produktes unabdingbar sind. Ohne ihre Umsetzung geht einfach nichts, deshalb müssen sie erfüllt werden.

**Wunschkriterien** Die nächste Kategorie, die Wunschkriterien, sind zwar entbehrlich, wurden aber ausdrücklich vom Auftraggeber gefordert. Ihre Umsetzung im Produkt ist also genauso eine Pflicht, wie die der ersten Kategorie, das Produkt würde aber auch ohne ihre Implementierung seinen Betrieb korrekt ausführen.

**Abgrenzungskriterien** Die letzte Kategorie beschreiben die Abgrenzungs-

kriterien. Sie sollen die Grenzen des Produktes klarmachen und sind von daher beinahe wichtiger als die beiden ersten Fälle denn sie hindern die Entwickler daran, über alle Ziele hinauszuschießen.

<https://st.inf.tu-dresden.de/SalesPoint/v3.2/tutorial/stepbystep2/pflh.html>

(b) Nennen Sie drei Einsatzziele von Softwaremaßen.

Lösungsvorschlag

Aus Baumann K. (1997) Softwaremaße. In: Unterstützung der objektorientierten Systemanalyse durch Softwaremaße. Beiträge zur Wirtschaftsinformatik, vol 23. Physica, Heidelberg. [https://doi.org/10.1007/978-3-662-13273-9\\_2](https://doi.org/10.1007/978-3-662-13273-9_2)

Der Einsatz von Softwaremaßen kann vielfältige Vorteile mit sich bringen. Folgende Ziele können mit der Anwendung von Softwaremaßen verfolgt werden:

- Durch eine Bewertung kann überprüft werden, inwieweit einmal gestellte *Qualitätsanforderungen* erfüllt wurden.
- Eine Bewertung bereits erstellter oder noch zu erstellender Softwareprodukte kann Hilfestellung bei Entscheidungen, die für die *Projektplanung* oder das Entwicklungsmanagement zu treffen sind, leisten. So können Softwaremaße zur Einschätzung des notwendigen Aufwands, der zu erbringenden Kosten und des Personalbedarfs herangezogen werden.
- Ein möglichst frühzeitiges *Aufzeigen von Schwachstellen im Systemdesign*, die Beurteilung der Auswirkungen neuer Techniken oder Tools auf die Produktivität der Entwickler oder auf die Qualität des entwickelten Produkts sowie die Überprüfung der Einhaltung festgelegter Designstandards sind weitere mögliche Einsatzfelder für Softwaremaße.
- Darüber hinaus ist der Einsatz von Softwaremaßen *Teil umfassender Konzepte zum Softwarequalitätsmanagement*. Softwaremaße bzw. die zugehörige Meßergebnisse tragen zunächst dazu bei, die betrachteten Produkte oder Prozesse besser zu verstehen oder hinsichtlich interessierender Eigenschaften zu bewerten.

[https://link.springer.com/chapter/10.1007%2F978-3-662-13273-9\\_2](https://link.springer.com/chapter/10.1007%2F978-3-662-13273-9_2)

(c) Welche Arten von Softwaremaßen werden unterschieden?

Lösungsvorschlag

Die Unterscheidung nach dem Messgegenstand ergibt folgende Arten von Maßen:

**Externe Produktmaße.** Diese beschreiben Merkmale des Produkts, die von außen sichtbar sind, wenngleich eventuell nur indirekt. Dazu gehören



insbesondere Maße für Qualitätsattribute wie die Wartbarkeit, die Zuverlässigkeit, die Benutzbarkeit, die Effizienz etc.

**Interne Produktmaße.** Basis für die Charakterisierung externer Produktattribute sind interne Attribute, die sich zumindest zum Teil besser messen lassen. Hierzu gehören z. B. die Größe, die Modularität und die Komplexität.

**Prozessmaße.** Diese charakterisieren Eigenschaften des Produktionsprozesses, z. B. die typische Produktivität, Kostenaufteilung auf bestimmte Arbeiten, Dauer von Arbeitsschritten etc.

**Ressourcenmaße.** Diese charakterisieren Eigenschaften der im Prozess verwendeten Ressourcen, z. B. die Auslastung von Rechnern, die typische Produktivität und Fehlerrate eines bestimmten Ingenieurs etc.

<http://page.mi.fu-berlin.de/prechelt/swt2/node5.html>

Was sind die 3 Arten der (einfachen) Maße zur Messung von Software?

<https://quizlet.com/de/339799466/softwaretechnik-theorie-flash-cards/>

- Größenorientiert/nach Größe
- Funktionsorientiert/nach Funktion
- Objektorientiert/nach Objekt

(d) Nennen Sie drei Merkmale evolutionärer Softwareentwicklung.

Lösungsvorschlag

Wie für jede Form der Software-Entwicklung stellt sich die Frage nach einem geeigneten methodischen Ansatz. Generell lässt sich der Prozess der Software-Entwicklung in die folgenden drei Abschnitte gliedern:

- (i) von der Problembeschreibung bis zur Software-Spezifikation (auch Problemanalyse, Systemanalyse, Requirements Engineering bzw. frühe Phasen der Software-Entwicklung genannt),
- (ii) die Software-Entwicklung im engeren (technischen) Sinn,
- (iii) die Integration der Software in den Anwendungszusammenhang.

Diese Einteilung gilt ebenso für die evolutionäre Software-Entwicklung, allerdings wird hier der Schwerpunkt anders gelegt. Steht beim klassischen Ansatz in der Regel die Software-Entwicklung im engeren Sinn im Mittelpunkt, so werden beim evolutionären Ansatz alle drei Abschnitte möglichst gleichgewichtig und im Zusammenhang betrachtet. Daraus resultiert eine natürliche Tendenz zu einer zyklischen Vorgehensweise.

Daneben gibt es weitere Querbezüge: Orientiert sich die Software-Entwicklung an einem festen Ziel, verfolgt dieses jedoch durch schrittweises Erweitern zunächst unvollkommener Teillösungen, so wird dieses Vorgehen inkrementell

genannt. Unfertige Versuchsanordnungen oder Teilsysteme, die am Beginn einer solchen Entwicklung stehen, werden oft als Prototypen bezeichnet. Wählt man Prototyping als Vorgehensweise, so kann entweder das spätere Produkt inkrementell aus einem oder mehreren Prototypen weiterentwickelt werden oder man setzt nach einer (Wegwerf-) Prototypphase neu auf. Beides sind Formen evolutionärer Entwicklung.

<http://waste.informatik.hu-berlin.de/~dahme/edidahm.pdf>

(e) Nennen Sie drei Vorteile des Einsatzes von Versionsverwaltungssoftware.

Lösungsvorschlag

- (i) Möglichkeit, auf ältere Entwicklungsversionen zurückzukehren, wenn sich die aktuelle als nicht lauffähig bzw. unsicher erwiesen hat (z. B. git revert)
- (ii) Möglichkeit, dass mehrere Entwickler gleichzeitig an ein- und demselben Softwareprojekt arbeiten und Möglichkeit, dass ihre Entwicklungen durch das Versionsverwaltungssystem zusammengeführt werden können (z. B. git merge)
- (iii) Möglichkeit, den Zeitpunkt oder den Urheber eines Softwarefehlers ausfindig zu machen (z. B. git blame)

(f) Worin besteht der Unterschied zwischen funktionalen und nicht-funktionalen Anforderungen?

Lösungsvorschlag

Funktionale Anforderung: Anforderung an die Funktionalität des Systems, also „Was kann es?“

Nicht-funktionale Anforderung: Design, Programmiersprache, Performanz

(g) Was verbirgt sich hinter dem Begriff „Continuous Integration“?

Lösungsvorschlag

Continuous Integration: Das fertige Modul wird sofort in das bestehende Produkt integriert. Die Integration erfolgt also schrittweise und nicht erst, wenn alle Module fertig sind. Somit können auch neue Funktionalitäten sofort hinzugefügt werden (*rightarrow* neue Programmversion).

## 66116 / 2016 / Herbst / Thema 1 / Teilaufgabe 2 / Aufgabe 2

(a) Gegeben sei folgende natürlichsprachliche Spezifikation eines PKI-Systems:

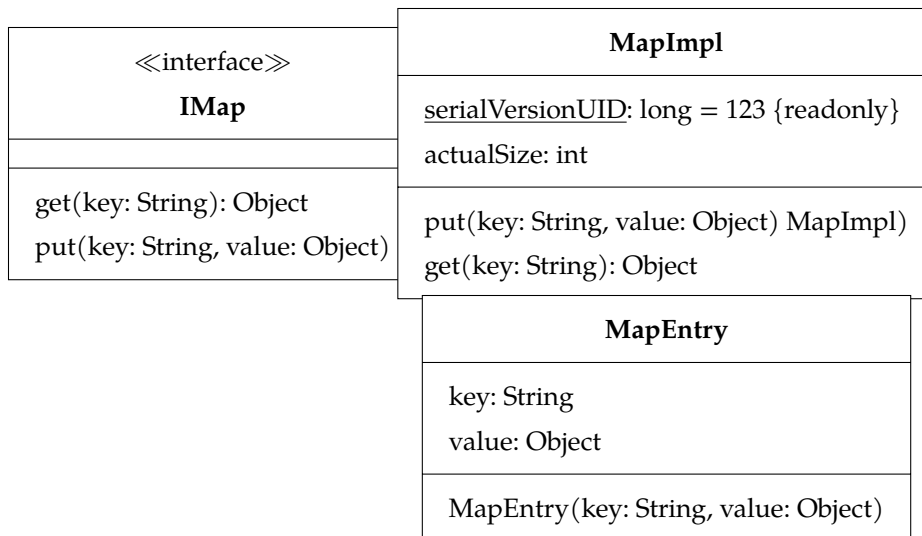
*Damit der Schüler seinem Lehrer die Hausaufgaben verschlüsselt per E-Mail übermitteln kann, bedarf es einer entsprechenden Infrastruktur. Nachdem beide Teilnehmer die notwendige Software installiert haben, erstellt der Lehrer zunächst ein sogenanntes Schlüsselpaar, bestehend aus einem „Öffentlichen“ (ÖS) und einem zugehörigen „privaten“*

Schlüssel (PS). Anschließend veröffentlicht der Lehrer seinen ÖS durch Hochladen auf einen sogenannten Keyserver (Schlüsselverzeichnisdienst). Damit steht er jedem Schüler zur Verfügung, so dass dieser den ÖS jederzeit vom Keyserver herunterladen kann. Alternativ kann der Lehrer seinen ÖS auch direkt (z. B. per E-Mail oder USB-Stick) an den Schüler übermitteln.

Der Schüler kann nun seine Nachricht (z. B. seine Lösung) mit dem ÖS des Lehrers verschlüsseln und mit einer E-Mail versenden. Empfängt der Lehrer eine solche E-Mail, so kann er (und nur er) mit seinem PS die Nachricht wieder entschlüsseln. Umgekehrt kann der Lehrer mit seinem PS beliebige Informationen (z. B. die Note) „digital signieren“. Diese unterschriebenen Daten übermittelt der Lehrer dann zusammen mit der digitalen Signatur per E-Mail an den Schüler. Der Schüler kann mit dem ÖS des Lehrers prüfen, ob die übermittelte Nachricht unverändert und tatsächlich vom unterschreibenden Lehrer stammt.

Modellieren Sie die in der Spezifikation beschriebenen Anwendungsfälle zusammen mit den jeweils beteiligten Akteuren in einem Use-Case-Diagramm. Betrachten Sie den Keyserver zunächst ebenfalls als Akteur, welcher gegenüber PKI als „externer Vermittler“ auftritt.

- (b) Erstellen Sie ein geeignetes Klassendiagramm für das obige PKI. Berücksichtigen Sie zusätzlich zur verbalen Spezifikation noch folgende Präzisierungen:
- (i) Die Schlüssel werden mit einer E-Mail-Adresse „benannt“, damit der Schüler den richtigen Schlüssel abrufen kann.
  - (ii) Es gibt genau einen Keyserver; dieser verwaltet aber beliebig viele Schlüssel. Er bietet die entsprechenden Dienste zum Veröffentlichen bzw. Abfragen von Schlüsseln an.
  - (iii) Jeder Lehrer hat höchstens ein Schlüsselpaar, aber jeder Schlüssel gehört genau einem Lehrer. Der Schüler hingegen kommuniziert mit mehreren Lehrern und kennt daher mehrere E-Mail-Adressen.
  - (iv) Eine Nachricht kann (muss aber nicht) eine Signatur oder einen ÖS als „Anhang“ zusätzlich zum eigentlichen Inhalt (zur Vereinfachung: String) mit sich führen.
  - (v) Für das Signieren bzw. Entschlüsseln ist der PS (das zugehörige Objekt selbst) zuständig. Dafür bekommt er den Inhalt der Nachricht und gibt entsprechend eine Signatur bzw. den entschlüsselten Inhalt zurück.
  - (vi) Für das Prüfen der Signatur und das Verschlüsseln ist der ÖS zuständig. Dazu bekommen die Methoden je nach Bedarf den Inhalt der Nachricht und die Signatur und liefern einen Wahrheitswert bzw. den verschlüsselten Inhalt zurück.
- (c) Übertragen Sie folgendes UML-Klassendiagramm in Programm-Code einer gängigen und geeigneten objektorientierten Sprache Ihrer Wahl. Die Methodenrumpfe dürfen Sie dabei leer lassen (auch wenn das Programm dann nicht übersetzbar bzw. ausführbar wäre).



Lösungsvorschlag

### Interface „IMap“

```
interface IMap {
    Object get(String key);

    void put(String key, Object value);
}
```

Code-Beispiel auf Github ansehen: [src/main/java/org/bschlangaul/examen/examen\\_66116/jahr\\_2016/herbst/pki/IMap.java](https://github.com/bschlangaul/examen/examen_66116/jahr_2016/herbst/pki/IMap.java)

### Klasse „MapImpl“

```
import java.util.ArrayList;
import java.util.List;

class MapImpl implements IMap {
    final static long serialVersionUID = 123;

    private List<MapEntry> entries;

    MapImpl() {
        entries = new ArrayList<MapEntry>();
    }

    public Object get(String key) {
        return new Object();
    }

    public void put(String key, Object value) {
        // Nicht verlangt in der Aufgabenstellung.
        entries.add(new MapEntry(key, value));
    }
}
```

Code-Beispiel auf Github ansehen: [src/main/java/org/bschlangaul/examen/examen\\_66116/jahr\\_2016/herbst/pki/MapImpl.java](https://github.com/bschlangaul/examen/examen_66116/jahr_2016/herbst/pki/MapImpl.java)Datenflussorientiertes Testen  
Bubblesort  
Datenfluss-annotierter  
Kontrollflussgraph**Klasse „MapEntry“**

```
class MapEntry {
    String key;
    Object value;

    MapEntry(String key, Object value) {
        // Nicht verlangt in der Aufgabenstellung.
        this.key = key;
        this.value = value;
    }
}
```

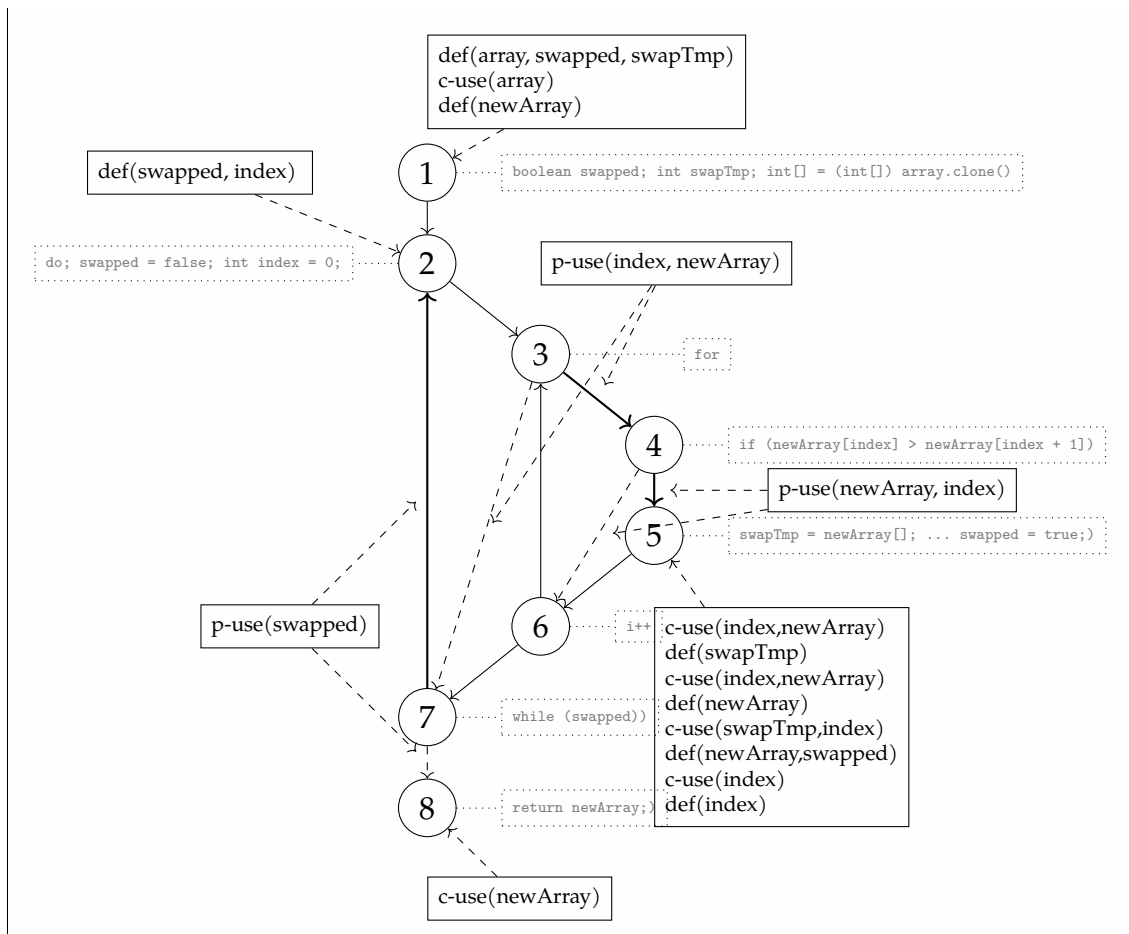
Code-Beispiel auf Github ansehen: [src/main/java/org/bschlangaul/examen/examen\\_66116/jahr\\_2016/herbst/pki/MapEntry.java](https://github.com/bschlangaul/examen/examen_66116/jahr_2016/herbst/pki/MapEntry.java)**66116 / 2016 / Herbst / Thema 1 / Teilaufgabe 2 / Aufgabe 3**Gegeben Sei folgende Java-Methode `sort` zum Sortieren eines Feldes ganzer Zahlen:

```
public static int[] sort(int[] array) {
    boolean swapped;
    int swapTmp;
    int[] newArray = (int[]) array.clone();
    do {
        swapped = false;
        for (int index = 0; index < newArray.length - 1; index++) {
            if (newArray[index] > newArray[index + 1]) {
                swapTmp = newArray[index];
                newArray[index] = newArray[index + 1];
                newArray[index + 1] = swapTmp;
                swapped = true;
            }
        }
    } while (swapped);
    return newArray;
}
```

Code-Beispiel auf Github ansehen: [src/main/java/org/bschlangaul/examen/examen\\_66116/jahr\\_2016/herbst/BubbleSort.java](https://github.com/bschlangaul/examen/examen_66116/jahr_2016/herbst/BubbleSort.java)

- (a) Konstruieren Sie den Kontrollflussgraphen des obigen Code-Fragments und annotieren Sie an den Knoten und Kanten die zugehörigen Datenflussinformationen (Definitionen bzw. berechnende oder prädikative Verwendung von Variablen).

Lösungsvorschlag



Zyklomatische Komplexität  
nach McCabe  
C1-Test Zweigüberdeckung  
(Branch Coverage)  
all uses

- (b) Nennen Sie die maximale Anzahl linear unabhängiger Programmpfade, also die zyklomatische Komplexität nach McCabe.

Lösungsvorschlag

Der Graph hat 8 Knoten und 10 Kanten. Daher ist die zyklomatische Komplexität nach McCabe gegeben durch  $10 - 8 + 2 = 4$ .

- (c) Geben Sie einen möglichst kleinen Testdatensatz an, der eine 100%-ige Verzweigungsüberdeckung dieses Moduls erzielt.

Lösungsvorschlag

Die Eingabe muss mindestens ein Feld der Länge 3 sein. Ansonsten wäre das Feld schon sortiert bzw. bräuchte nur eine Vertauschung und die innere if-Bedingung wäre nicht zu 100% überdeckt. Daher wählt man beispielsweise  $\text{array} = [1, 3, 2]$ .

- (d) Beschreiben Sie kurz, welche Eigenschaften eine Testfallmenge allgemein haben muss, damit das datenflussorientierte Überdeckungskriterium „all-uses“ erfüllt.

Das Kriterium all-uses ist das Hauptkriterium des datenflussorientierten Testens, denn es testet den kompletten prädikativen und berechnenden Datenfluss. Konkret: von jedem Knoten mit einem globalen  $\text{def}(x)$  einer Variable  $x$

existiert ein definitions-freier Pfad bzgl.  $x$  ( $\text{def-clear}(x)$ ) zu jedem erreichbaren Knoten mit einem  $\text{c-use}(x)$  oder  $\text{p-use}(x)$ .

## 66116 / 2016 / Herbst / Thema 2 / Teilaufgabe 2 / Aufgabe 1

Ordnen Sie die folgenden Aussagen entsprechend ihres Wahrheitsgehaltes in einer Tabelle der folgenden Form an:

Kategorie	WAHR	FALSCH
X	X1, X3	X2
Y	Y2	Y1
...	...	...

### A Allgemein

- A1** Im Software Engineering geht es vor allem darum qualitativ hochwertige Software zu entwickeln.
- A2** Software Engineering ist gleichbedeutend mit Programmieren.

### B Vorgehensmodelle

- B1** Die Erhebung und Analyse von Anforderungen sind nicht Teil des Software Engineerings.
- B2** Agile Methoden eignen sich besonders gut für die Entwicklung komplexer und sicherer Systeme in verteilten Entwicklerteams.
- B3** Das Spiralmodell ist ein Vorläufer sogenannter Agiler Methoden.

### C Anforderungserhebung

- C1** Bei der Anforderungserhebung dürfen in keinem Fall mehrere Erhebungstechniken (z. B. Workshops, Modellierung) angewendet werden, weil sonst Widersprüche in Anforderungen zu, Vorschein kommen könnten.
- C2** Ein Szenario beinhaltet eine Menge von Anwendungsfällen.
- C3** Nicht-funktionale Anforderungen sollten, wenn möglich, immer quantitativ spezifiziert werden.

### D Architekturmuster

- D1** Schichtenarchitekturen sind besonders für Anwendungen geeignet, in denen Performance eine wichtige Rolle spielt.
- D2** Das Black Board Muster ist besonders für Anwendungen geeignet, in denen Performance eine wichtige Rolle spielt.



**D3** „Dependency Injection“ bezeichnet das Konzept, welches Abhängigkeiten zur Laufzeit reglementiert.

Einbringen von  
Abhängigkeiten  
(Dependency Injection)  
Sequenzdiagramm  
Zustandsdiagramm Wissen  
Komponentendiagramm  
Modell-Präsentation-  
Steuerung  
(Model-View-Controller)  
Einzelstück (Singleton)  
Kommando (Command)  
Validation  
Verifikation

## E UML

**E1** Sequenzdiagramme beschreiben Teile des Verhaltens eines Systems.

**E2** Zustandsübergangsdiagramme beschreiben das Verhalten eines Systems.

**E3** Komponentendiagramme beschreiben die Struktur eines Systems.

## F Entwurfsmuster

**F1** Das MVC Pattern verursacht eine starke Abhängigkeit zwischen Datenmodell und Benutzeroberfläche.

**F2** Das Singleton Pattern stellt sicher, dass es zur Laufzeit von einer bestimmten Klasse höchstens ein Objekt gibt.

**F3** Im Kommando Entwurfsmuster (engl. „Command Pattern“) werden Befehle in einem sog. Kommando-Objekt gekapselt, um sie bei Bedarf rückgängig zu machen.

## G Testen

**G1** Validation dient der Überprüfung von Laufzeitfehlern.

**G2** Testen ermöglicht sicherzustellen, dass ein Programm absolut fehlerfrei ist.

**G3** Verifikation dient der Überprüfung, ob ein System einer Spezifikation entspricht.

Lösungsvorschlag

Kategorie	WAHR	FALSCH
A	A1	A2
B	B3	B1, B2
C	C3	C1, C2
D	D3	D1, D2
E	E1, E2, E3	
F	F2, F3	F1
G	G3	G1 <sup>a</sup> , G2 <sup>b</sup>

<sup>a</sup>Validierung: Prüfung der Eignung beziehungsweise der Wert einer Software bezogen auf ihren Einsatzzweck: „Wird das richtige Produkt entwickelt?“

<sup>b</sup>Ein Softwaretest prüft und bewertet Software auf Erfüllung der für ihren Einsatz definierten Anforderungen und misst ihre Qualität.

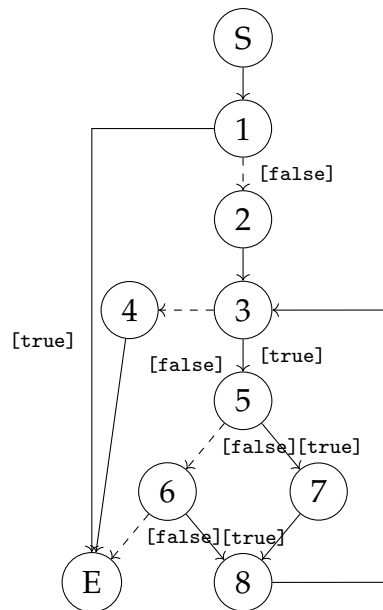
## 66116 / 2017 / Frühjahr / Thema 2 / Teilaufgabe 2 / Aufgabe 1

Gegeben Sei folgende Methode und ihr Kontrollflussgraph:

```

int binToInt(String bin) {
    if (bin.isEmpty())
        return -1;
    int place = 1, value = 0;
    int length = bin.length() -
    ↪ 1;
    ↪ for (int i = length; i >= 0;
    ↪ --i) {
        char ch = bin.charAt(i);
        if (ch == '1') {
            value += place;
        } else if (ch == '0') {
            // do nothing
        } else {
            return -1;
        }
        place *= 2;
    }
    return value;
}

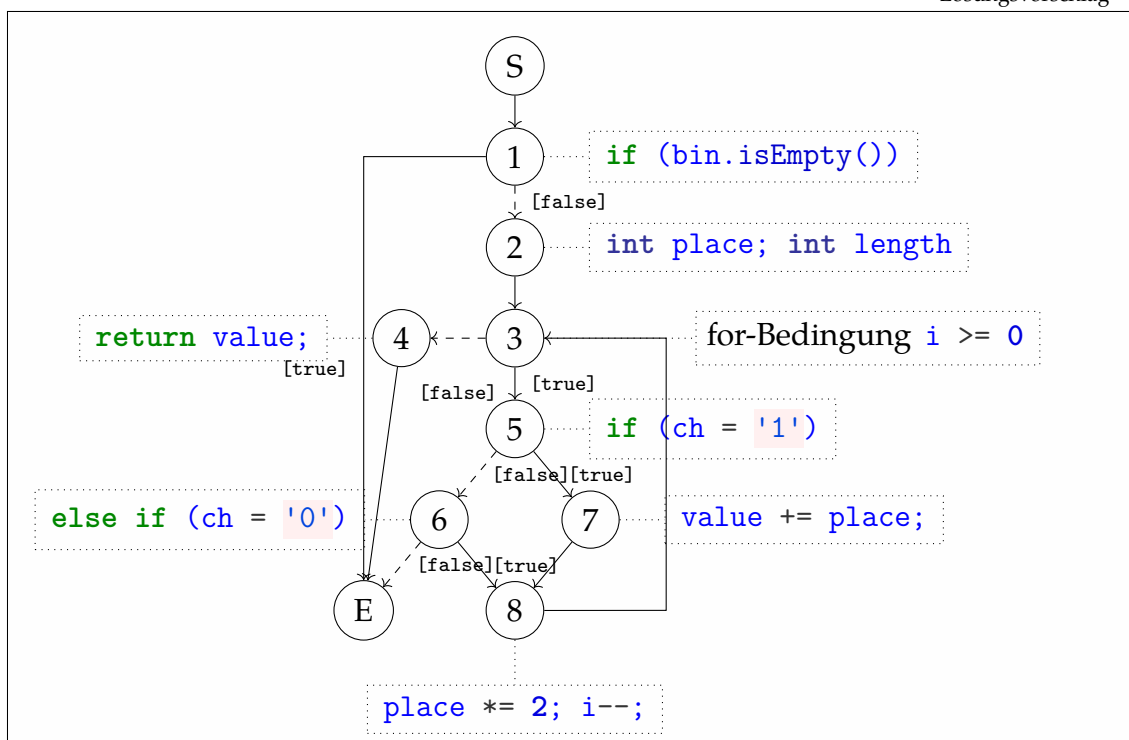
```



Code-Beispiel auf Github ansehen: [src/main/java/org/bschlangaul/aufgaben/sosy/ab\\_7/Aufgabe5.java](https://github.com/bschlangaul/aufgaben/sosy/ab_7/Aufgabe5.java)

- (a) Geben Sie je einen Repräsentanten aller Pfadklassen im Kontrollflussgraphen an, die zum Erzielen einer vollständigen

Lösungsvorschlag



- (i) Verzweigungsüberdeckung

```
p1 (Pfad 1) S 1 E
p2 S 1 2 3 4 E
p3 S 1 2 3 5 7 8 3 5 6 8 3 5 6 E
```

## (ii) Schleife-Inneres-Überdeckung

**Äußere Pfade (äußere Pfade):** (ohne Ausführung der Wiederholung)

```
p1 S 1 E
p2 S 1 2 3 4 E
```

**Grenzpfade (boundary test)** (alle Pfade, die die Wiederholung betreten, aber nicht wiederholen; innerhalb des Schleifenrumpfes alle Pfade!)

```
p4 S 1 2 3 5 6 E
```

**Innere Pfade (interior test)** (alle Pfade mit *einer Wiederholung des Schleifenrumpfes*; innerhalb des Schleifenrumpfes wieder alle Pfade!)

```
p5 S 1 2 3 5 7 8 3 5 7 8 3 4 E
p6 S 1 2 3 5 7 8 3 5 6 8 3 4 E
p7 S 1 2 3 5 6 8 3 5 6 8 3 4 E
p8 S 1 2 3 5 6 8 3 5 7 8 3 4 E
p9 S 1 2 3 5 7 8 3 5 7 8 3 5 6 E
p10 = p3 S 1 2 3 5 7 8 3 5 6 8 3 5 6 E
p11 S 1 2 3 5 6 8 3 5 6 8 3 5 6 E
p12 S 1 2 3 5 6 8 3 5 7 8 3 5 6 E
```

mit minimaler Testfallanzahl genügen würden.

- (b) Welche der vorangehend ermittelten Pfade sind mittels Testfälle tatsächlich überdeckbar („feasible“)? Falls der Pfad ausführbar ist, geben Sie bitte den Testfall an, andernfalls begründen Sie kurz, weshalb der Pfad nicht überdeckbar ist.

Erweitere Methode, die die Knotennamen ausgibt:

```
public static final String RESET = "\u001B[0m";
public static final String ROT = "\u001B[31m";
public static final String GRÜN = "\u001B[32m";

static int binToIntLog(String bin) {
    System.out.println("\nInput: " + bin);
    System.out.print("S");
    System.out.print(1);
    if (bin.isEmpty()) {
        System.out.print("E");
        System.out.println("\nOutput: " + -1);
    }
}
```

```
        return -1;
    }
    System.out.print(2);
    int place = 1, value = 0;
    int length = bin.length() - 1;
    System.out.print(3);
    for (int i = length; i >= 0; --i) {
        char ch = bin.charAt(i);
        System.out.print(5);
        if (ch == '1') {
            System.out.print(ROT + 7 + RESET);
            value += place;
        } else {
            System.out.print(GRÜN + 6 + RESET);
            if (ch == '0') {
                // do nothing
            } else {
                System.out.print("E");
                System.out.println("\nOutput: " + -1);
                return -1;
            }
        }
    }
    System.out.print(8);
    place *= 2;
    System.out.print(3);
}
System.out.print(4);
System.out.print("E");
System.out.println("\nOutput: " + value);
return value;
}

public static void main(String[] args) {
    binToIntLog(""); // p1
    binToIntLog("??"); // p2 not feasible
    binToIntLog("x01"); // p3
    binToIntLog("x"); // p4

    binToIntLog("11"); // p5
    binToIntLog("01"); // p6
    binToIntLog("00"); // p7
    binToIntLog("10"); // p8

    binToIntLog("x11"); // p9
    binToIntLog("x01"); // p10
    binToIntLog("x00"); // p11
    binToIntLog("x10"); // p12
}
}
```

Code-Beispiel auf Github ansehen: [src/main/java/org/bschlangaul/aufgaben/sosy/ab\\_7/Aufgabe5.java](https://github.com/bschlangaul/aufgaben/sosy/ab_7/Aufgabe5.java)

Alle mit Ausnahme von p2.

**p2** ist nicht überdeckbar. Passiert ein Wert der Variable **bin** die erste if-Verzweigung, dann hat der Wert eine Länge größer 0 und betritt deshalb die Wiederholung mit fester Anzahl.

p1	S 1 E	<code>binToInt("");</code>
p2	S 1 2 3 4 E	not feasible
p3	S 1 2 3 5 7 8 3 5 6 8 3 5 6 E	<code>binToInt("x01");</code>
p4	S 1 2 3 5 6 E	<code>binToInt("x");</code>
p5	S 1 2 3 5 7 8 3 5 7 8 3 4 E	<code>binToInt("11");</code>
p6	S 1 2 3 5 7 8 3 5 6 8 3 4 E	<code>binToInt("01");</code>
p7	S 1 2 3 5 6 8 3 5 6 8 3 4 E	<code>binToInt("00");</code>
p8	S 1 2 3 5 6 8 3 5 7 8 3 4 E	<code>binToInt("10");</code>
p9	S 1 2 3 5 7 8 3 5 7 8 3 5 6 E	<code>binToInt("x11");</code>
p10 = p3	S 1 2 3 5 7 8 3 5 6 8 3 5 6 E	<code>binToInt("x01");</code>
p11	S 1 2 3 5 6 8 3 5 6 8 3 5 6 E	<code>binToInt("x00");</code>
p12	S 1 2 3 5 6 8 3 5 7 8 3 5 6 E	<code>binToInt("x10");</code>

- (c) Bestimmen Sie anhand des Kontrollflussgraphen die maximale Anzahl linear unabhängiger Programmpfade, also die zyklomatische Komplexität nach Mc-Cabe.

Lösungsvorschlag

**Binärverzweigungen** 4

**Knoten** 10

**Kanten** 13

Anhand der Binärverzweigungen:

$$\begin{aligned}
 M &= b + p \\
 &= 4 + 1 \\
 &= 5
 \end{aligned}$$

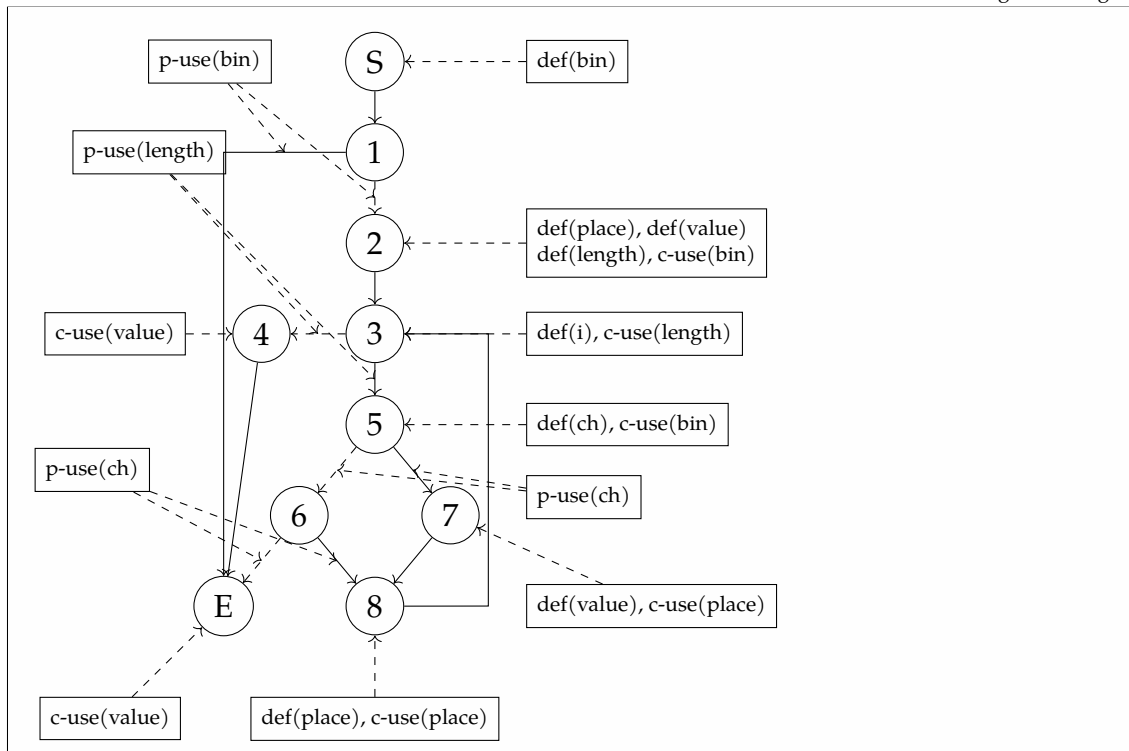
oder durch Anzahl Kanten  $e$  und Knoten  $n$

$$\begin{aligned}
 M &= e - n + 2p \\
 &= 13 - 10 + 2 \cdot 1 \\
 &= 5
 \end{aligned}$$

- (d) Kann für dieses Modul eine 100%-ige Pfadüberdeckung erzielt werden? Begründen Sie kurz Ihre Antwort.

Nein, da **p2** nicht überdeckbar ist.

- (e) Geben Sie zu jedem Knoten die jeweilige Datenfluss-Annotation (defs bzw. uses) für jede betroffene Variable in der zeitlichen Reihenfolge ihres Auftretens zur Laufzeit an.



## 66116 / 2017 / Frühjahr / Thema 2 / Teilaufgabe 2 / Aufgabe 4

Sie dürfen im Folgenden davon ausgehen, dass keinerlei Under- oder Overflows auftreten.

Gegeben sei die folgende Methode mit Vorbedingung  $P := x \geq 0 \wedge y \geq 0$  und Nachbedingung  $Q := x \cdot y = z$ .

```
int mul (int x , int y) {
    /* P */
    int z = 0, i = 0;
    while (i++ != x)
        z += y;
    /* Q */
    return z;
}
```

Betrachten Sie dazu die folgenden drei Prädikate:

$$- I_1 := z + i \cdot y = x \cdot y$$

-  $I_2 := \text{false}$

-  $I_3 := z + (x - i) \cdot y = x \cdot y$

- (a) Beweisen Sie formal für jedes der drei Prädikate, ob es unmittelbar vor Betreten der Schleife in `mul` gilt oder nicht.

Lösungsvorschlag

$$\begin{aligned} \text{wp}(\text{"Code vor der Schleife"}, I_1) &\equiv \text{wp}(\text{"int } z = 0, i = 0;", z + i \cdot y = x \cdot y) \\ &\equiv \text{wp}("", 0 + 0 \cdot y = x \cdot y) \\ &\equiv 0 = x \cdot y \\ &\equiv \text{falsch} \end{aligned}$$

$$\begin{aligned} \text{wp}(\text{"Code vor der Schleife"}, I_2) &\equiv \text{wp}(\text{"int } z = 0, i = 0;", \text{false}) \\ &\equiv \text{wp}("", \text{false}) \\ &\equiv \text{false} \\ &\equiv \text{falsch} \end{aligned}$$

$$\begin{aligned} \text{wp}(\text{"Code vor der Schleife"}, I_3) &\equiv \text{wp}(\text{"int } z = 0, i = 0;", z + (x - i) \cdot y = x \cdot y) \\ &\equiv \text{wp}("", 0 + (x - 0) \cdot y = x \cdot y) \\ &\equiv x \cdot y = x \cdot y \\ &\equiv \text{wahr} \end{aligned}$$

- (b) Weisen Sie formal nach, welche der drei Prädikate Invarianten des Schleifenrumpfs in `mul` sind oder welche nicht.

Lösungsvorschlag

Für den Nachweis muss der Code etwas umformuliert werden:

```
int mul (int x , int y) {
    /* P */
    int z = 0, i = 0;
    while (i != x) {
        i = i + 1;
        z = z + y;
    }
    /* Q */
    return z;
}
```

$$\begin{aligned}
\text{wp}(\text{"Code Schleife"}, I_1 \wedge i \neq x) &\equiv \text{wp}(\text{"i = i + 1; z = z + y;"}, z + i \cdot y = x \cdot y \wedge i \neq x) \\
&\equiv \text{wp}(\text{"i = i + 1;"}, z + y + i \cdot y = x \cdot y \wedge i \neq x) \\
&\equiv \text{wp}(\text{""}, z + y + (i + 1) \cdot y = x \cdot y \wedge i + 1 \neq x) \\
&\equiv z + y + (i + 1) \cdot y = x \cdot y \wedge i + 1 \neq x \\
&\equiv z + i \cdot y + 2 \cdot y = x \cdot y \wedge i + 1 \neq x \\
&\equiv \text{falsch} \wedge i + 1 \neq x \\
&\equiv \text{falsch}
\end{aligned}$$

$$\begin{aligned}
\text{wp}(\text{"Code Schleife"}, I_2 \wedge i \neq x) &\equiv \text{wp}(\text{"i = i + 1; z = z + y;"}, \text{false} \wedge i \neq x) \\
&\equiv \text{wp}(\text{""}, \text{false} \wedge i \neq x) \\
&\equiv \text{falsch} \wedge i \neq x \\
&\equiv \text{falsch}
\end{aligned}$$

$$\begin{aligned}
\text{wp}(\text{"Code Schleife"}, I_3 \wedge i \neq x) &\equiv \text{wp}(\text{"i = i + 1; z = z + y;"}, z + (x - i) \cdot y = x \cdot y \wedge i \neq x) \\
&\equiv \text{wp}(\text{"i = i + 1;"}, z + y + (x - i) \cdot y = x \cdot y \wedge i \neq x) \\
&\equiv \text{wp}(\text{""}, z + y + (x - i + 1) \cdot y = x \cdot y \wedge i + 1 \neq x) \\
&\equiv z + y + x \cdot y - i \cdot y + y = x \cdot y \wedge i + 1 \neq x \\
&\equiv z + 2 \cdot y + x \cdot y - i \cdot y = x \cdot y \wedge i + 1 \neq x \\
&\equiv \text{wahr}
\end{aligned}$$

- (c) Beweisen Sie formal, aus welchen der drei Prädikate die Nachbedingung gefolgt werden darf bzw. nicht gefolgt werden kann.

Lösungsvorschlag

$$I_1 := z + i \cdot y = x \cdot y \quad I_2 := \text{false} \quad I_3 := z + (x - i) \cdot y = x \cdot y$$

$$\begin{aligned}
\text{wp}(\text{"Code nach Schleife"}, I_1 \wedge i = x) &\equiv \text{wp}(\text{""}, z + i \cdot y = x \cdot y \wedge i = x) \\
&\equiv z + i \cdot y = x \cdot y \wedge i = x \\
&\equiv z + x \cdot y = x \cdot y \\
&\equiv \text{wahr}
\end{aligned}$$



$$\begin{aligned}
 \text{wp}(\text{"Code nach Schleife"}, I_2 \wedge i = x) &\equiv \text{wp}(\text{"", false} \wedge i = x) \\
 &\equiv \text{false} \wedge i = x \\
 &\equiv \text{falsch} \\
 &\neq Q
 \end{aligned}$$

$$\begin{aligned}
 \text{wp}(\text{"Code nach Schleife"}, I_3 \wedge i = x) &\equiv \text{wp}(\text{"", } z + (x - i) \cdot y = x \cdot y \wedge i = x) \\
 &\equiv z + (x - i) \cdot y = x \cdot y \wedge i = x \\
 &\equiv z + (x - x) \cdot y = x \cdot y \\
 &\equiv z + 0 \cdot y = x \cdot y \\
 &\equiv z + 0 = x \cdot y \\
 &\equiv z = x \cdot y \\
 &\equiv Q
 \end{aligned}$$

- (d) Skizzieren Sie den Beweis der totalen Korrektheit der Methode `mul`. Zeigen Sie dazu auch die Terminierung der Methode.

Lösungsvorschlag

Aus den Teilaufgaben folgt der Beweis der partiellen Korrektheit mit Hilfe der Invariante  $i_3$ .  $i$  steigt streng monoton von 0 an so lange gilt  $i \neq x$ .  $i = x$  ist die Abbruchbedingung für die bedingte Wiederholung. Dann terminiert die Methode. Die Methode `mul` ist also total korrekt.

## 66116 / 2017 / Herbst / Thema 1 / Teilaufgabe 2 / Aufgabe 4

Die folgende Seite enthält Software-Quellcode, der einen Algorithmus zur binären Suche implementiert. Dieser ist durch Inspektion zu überprüfen. Im Folgenden sind die Regeln der Inspektion angegeben.

RM1	(Dokumentation)	Jede Quellcode-Datei beginnt mit einem Kommentar, der den Klassennamen, Versionsinformationen, Datum und Urheberrechtsangaben enthält.
RM2	(Dokumentation)	Jede Methode wird kommentiert. Der Kommentar enthält eine vollständige Beschreibung der Signatur so wie eine Design-by-Contract-Spezifikation.
RM3	(Dokumentation)	Deklarationen von Variablen werden kommentiert.
RM4	(Dokumentation)	Jede Kontrollstruktur wird kommentiert.
RM5	(Formatierung)	Zwischen einem Schlüsselwort und einer Klammer steht ein Leerzeichen.
RM6	(Formatierung)	Zwischen binären Operatoren und den Operanden stehen Leerzeichen.
RM7	(Programmierung)	Variablen werden in der Anweisung initialisiert, in der sie auch deklariert werden.
RM8	(Bezeichner)	Klassennamen werden groß geschrieben, Variablennamen klein.

```
/**
 * BinarySearch.java
 *
 * Eine Implementierung der "Binaere Suche"
 * mit einem iterativen Algorithmus
 */
class BinarySearch {

    /**
     * BinaereSuche
     * a: Eingabefeld
     * item: zuzuchendesElement
     * returnValue: der Index des zu suchenden Elements oder -1
     *
     * Vorbedingung:
     * a.length > 0
     * a ist ein linear geordnetes Feld:
     * For all k: (1 <= k < a.length) ==> (a[k-1] <=a [k])
     *
     * Nachbedingung:
     * Wenn item in a, dann gibt es ein k mit a[k] == item und returnValue == k
     * Genau dann wenn returnValue == -1 gibt es kein k mit 0 <= k < a.length
     * und a[k]==item.
     */
    public static int binarySearch(float a[], float item) {

        int End; // exklusiver Index fuer das Ende des
                // zuzuchsuchenden Teils des Arrays
    }
```

```
int start = 1; // inklusiver Index fuer den Anfang der Suche
End = a.length;

// Die Schleife wird verlassen, wenn keine der beiden Haelften das
// Element enthaelt.
while(start < End) {

    // Teilung des Arrays in zwei Haelften
    // untere Haelfte: [0,mid[
    // obere Haelfte: ]mid,End[
    int mid = (start + End) / 2;

    if (item > a[mid]) {
        // Ausschluss der oberen Haelfte
        start = mid + 1;
    } else if(item < a[mid]) {
        // Ausschluss der unteren Haelfte
        End = mid-1;
    } else {
        // Das gesuchte Element wird zurueckgegeben
        return (mid);
    }
} // end of while

// Bei Misserfolg der Suche wird -1 zurueckgegeben
return (-1);
}
```

- (a) Überprüfen Sie durch Inspektion, ob die obigen Regeln für den Quellcode eingehalten wurden. Erstellen Sie eine Liste mit allen Verletzungen der Regeln. Geben Sie für jede Verletzung einer Regel die Zeilennummer, Regelnummer und Kommentar an, z. B. (07, RM4, while nicht kommentiert). Schreiben Sie nicht in den Quellcode.

Lösungsvorschlag

--

Zeile	Regel	Kommentar
3-8	RM1	Fehlen von Versionsinformationen, Datum und Urheberrechtsangaben
11-26	RM2	Fehlen der Invariante in der Design-by-Contract-Spezifikation
36,46	RM5	Fehlen des Leerzeichens vor der Klammer
48	RM6	Um einen binären (zweistellige) Operator handelt es sich im Code-Beispiel um den Subtraktionsoperator: <code>mid-1</code> . Hier fehlen die geforderten Leerzeichen.
32	RM7	Die Variable <code>End</code> wird in Zeile 32 deklariert, aber erst in Zeile initialisiert <code>End = a.length;</code>
32	RM8	Die Variable <code>End</code> muss klein geschrieben werden.

- (b) Entspricht die Methode `binarySearch` ihrer Spezifikation, die durch Vor- und Nachbedingungen angegeben ist? Geben Sie gegebenenfalls Korrekturen der Methode an.

Lösungsvorschlag

### Korrektur der Vorbedingung

Die Vorbedingung ist nicht erfüllt, da weder die Länge des Feldes `a` noch die Reihenfolge der Feldeinträge geprüft wurden.

```

if (a.length <= 0) {
    return -1;
}

for (int i = 0; i < a.length; i++) {
    if (a[i] > a[i + 1]) {
        return -1;
    }
}

```

### Korrektur der Nachbedingung

`int start` muss mit `0` initialisiert werden, da sonst `a[0]` vernachlässigt wird.

- (c) Beschreiben alle Kommentare ab Zeile 24 die Semantik des Codes korrekt? Geben Sie zu jedem falschen Kommentar einen korrigierten Kommentar mit Zeilennummer an.

Zeile	Kommentar im Code	Korrektur
34-35	<code>// Die Schleife wird v erlassen, wenn keine der beiden Haelften das Elemen t enthaelt.</code>	<code>// Die Schleife wird v erlassen, wenn keine der beiden Haelften das El ement enthaelt oder das Element gefunden wurde.</code>
44	<code>// Ausschluss der oberen Haelfte</code>	<code>// Ausschluss der unteren Haelfte</code>
47	<code>// Ausschluss der unteren Haelfte</code>	<code>// Ausschluss der oberen Haelfte</code>
50	<code>// Das gesuchte Element wird zurueckgegeben</code>	<code>// Der Index des gesuchten Elements wird zurueckgeb en</code>

- (d) Geben Sie den Kontrollflussgraphen für die Methode `binarySearch` an.
- (e) Geben Sie maximal drei Testfälle für die Methode `binarySearch` an, die insgesamt eine vollständige Anweisungsüberdeckung leisten.

Die gegebene Methode: `binarySearch(a[], item)`

### Testfall

- (i) Testfall: `a[] = {1, 2, 3}, item = 4`
- (ii) Testfall: `a[] = {1, 2, 3}, item = 2`

## 66116 / 2017 / Herbst / Thema 2 / Teilaufgabe 2 / Aufgabe 1

Gegeben ist das folgende Gantt-Diagramm zur Planung eines hypothetischen Softwareprojekts:

- (a) Konvertieren Sie das Gantt-Diagramm in ein CPM-Netzwerk, das die Aktivitäten und Abhängigkeiten äquivalent beschreibt. Gehen Sie von der Zeiteinheit „Monate“ aus. Definieren Sie im CPM-Netzwerk je einen globalen Start- und Endknoten. Der Start jeder Aktivität hängt dabei vom Projektstart ab, das Projektende hängt vom Ende aller Aktivitäten ab.
- (b) Berechnen Sie für jedes Ereignis (öfür jeden Knoten Ihres CPM-Netzwerks) die früheste Zeit, die späteste Zeit sowie die Pufferzeit. Beachten Sie, dass die Berechnungsreihenfolge einer topologischen Sortierung des Netzwerks entsprechen sollte.
- (c) Geben Sie einen kritischen Pfad durch das CPM-Netzwerk an. Welche Aktivität darf sich demnach wie lange verzögern?

## 66116 / 2017 / Herbst / Thema 2 / Teilaufgabe 2 / Aufgabe 2

Aktivitätsdiagramm  
Klassendiagramm  
Objektdiagramm

Gegeben sei das folgende Glossar, welches die statische Struktur von einfachen Aktivitätsdiagrammen in natürlicher Sprache beschreibt:

**Aktivitätsdiagramm:** Benannter Container für Aktivitäten und Datenflüsse. Eine der definierten Aktivitäten ist als Start-Aktivität ausgezeichnet.

**Aktivität:** Teil des beschriebenen Verhaltens. Man unterscheidet Start-, End-, echte Aktivitäten sowie Entscheidungen. Aktivitäten können generell mehrere ein- und auslaufende Kontrollflüsse haben.

**Startaktivität:** Ist im Aktivitätsdiagramm eindeutig und dient als Einstiegspunkt des beschriebenen Ablaufs.

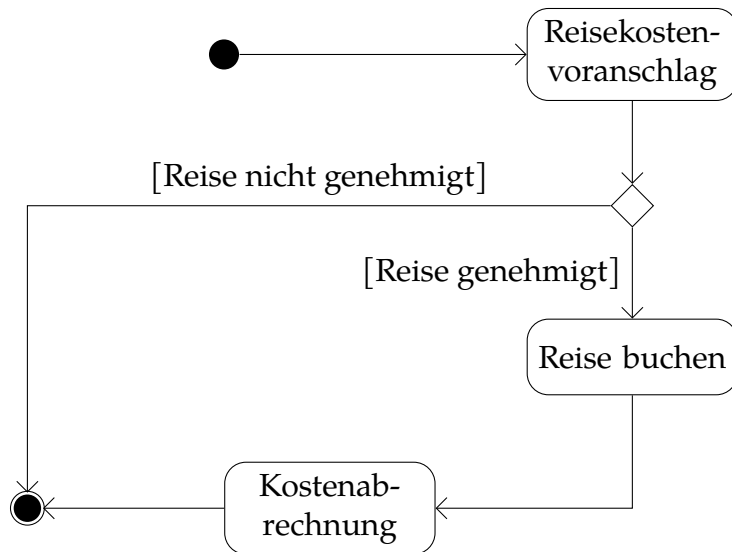
**Endaktivität:** Wird eine solche Aktivität erreicht, ist der beschriebene Ablauf zu Ende.

**Echte Aktivität:** Benannte Aktion, die nach Ausführung zu einer definierten nächsten Aktivität führt.

**Entscheidung:** Aktivität, die mehrere Nachfolger hat. Welche davon als nächstes ausgeführt wird, wird durch entsprechende Bedingungen (s. Kontrollfluss) gesteuert.

**Kontrollfluss:** Verbindet je eine Quell- mit einer Zielaktivität. Kann eine Bedingung enthalten, die erfüllt sein muss, damit die Zielaktivität im Falle einer Entscheidung ausgeführt wird.

- (a) Geben Sie ein UML-Klassendiagramm an, welches die im Glossar definierten Konzepte und Beziehungen formal beschreibt. Geben Sie bei allen Attributen und Assoziationsenden deren Sichtbarkeit, Multiplizität imd Typ an. Benennen Sie alle Assoziationen.
- (b) Nachfolgend ist ein Beispiel eines Aktivitätsdiagramms in der gängigen grafischen Notation abgebildet. Stellen Sie den beschriebenen Kontrollfluss als UML-Objektdiagramm konform zum in Teilaufgabe a erstellten UML-Klassendiagramm dar. Referenzieren Sie die dort definierten Klassen und Assoziationen; auf Objektbezeichner dürfen Sie verzichten.



## 66116 / 2018 / Frühjahr / Thema 2 / Teilaufgabe 2 / Aufgabe 1

Gegeben sei das folgende Java-Programm:

```
class M {
    private boolean b;
    private F f;
    private A a;

    public void m() {
        f = new F();
        a = new A(f);
        b = true;
    }
}

class A {
    private R r;
    public A(I i) {
        r = i.createX();
    }
}

interface I {
    public X createX();
}

class F implements I {
    public X createX() {
        return new X(0, 0);
    }
}

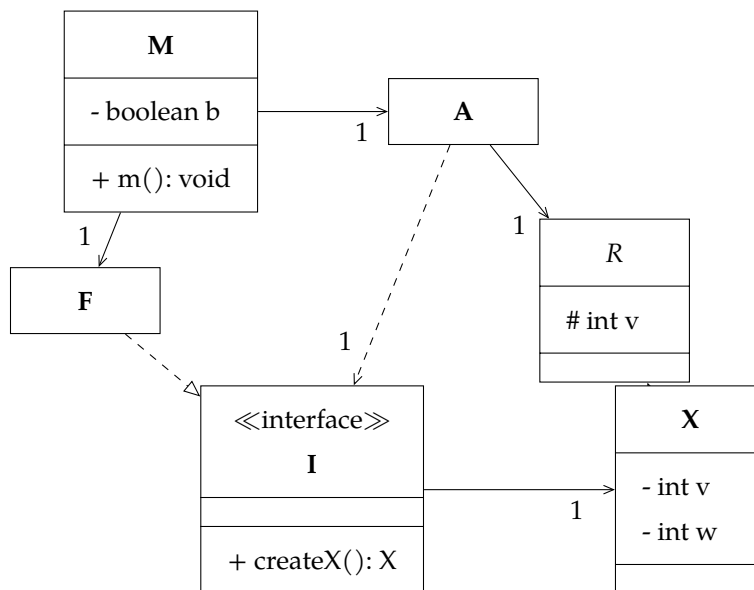
abstract class R {
    protected int v;
}
```

```

class X extends R {
    private int v, w;
    public X(int v, int w) {
        this.v = v;
        this.w = w;
    }
}

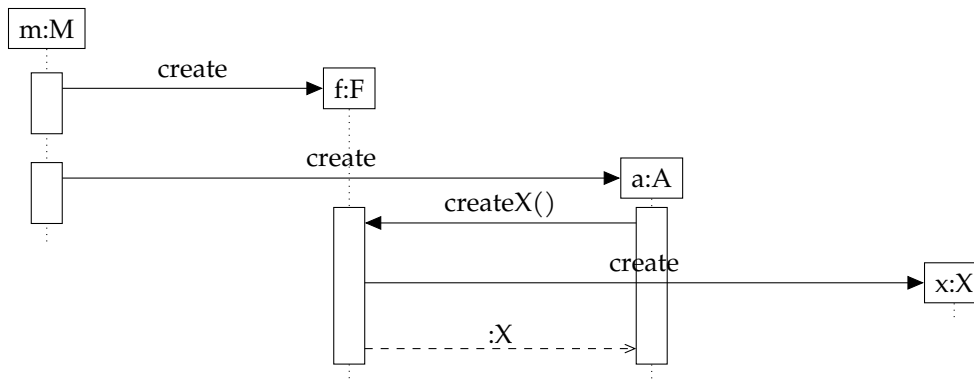
```

- (a) Das Subtypprinzip der objektorientierten Programmierung wird in obigem Programmcode zweimal ausgenutzt. Erläutern Sie wo und wie dies geschieht.
- (b) Zeichnen Sie ein UML-Klassendiagramm, das die statische Struktur des obigen Programms modelliert. Instanzvariablen mit einem Klassentyp sollen durch gerichtete Assoziationen mit Rollennamen und Multiplizität am gerichteten Assoziationsende modelliert werden. Alle aus dem Programmcode ersichtlichen statischen Informationen (insbesondere Interfaces, abstrakte Klassen, Zugriffsrechte, benutzerdefinierte Konstruktoren und Methoden) sollen in dem Klassendiagramm abgebildet werden.



- (c) Es wird angenommen, dass ein Objekt der Klasse M existiert, für das die Methode `m()` aufgerufen wird. Geben Sie ein Instanzendiagramm (Objektdiagramm) an, das alle nach der Ausführung der Methode `m` existierenden Objekte und deren Verbindungen (Links) zeigt.
- (d) Wie in Teil c) werde angenommen, dass ein Objekt der Klasse M existiert, für das die Methode `m()` aufgerufen wird. Diese Situation wird in Abb. 1 dargestellt. Zeichnen Sie ein Sequenzdiagramm, das Abb. 1 so ergänzt, dass alle auf den Aufruf der Methode `m()` folgenden Objekterzeugungen und Interaktionen gemäß der im Programmcode angegebenen Konstruktor- und Methodenrumpfe dargestellt werden. Aktivierungsphasen von Objekten sind durch längliche Rechtecke deutlich zu machen.

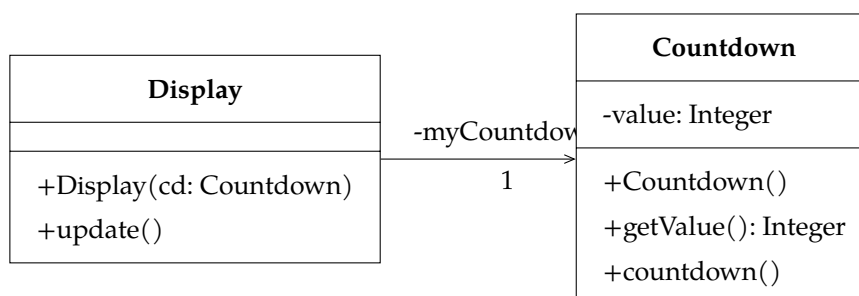




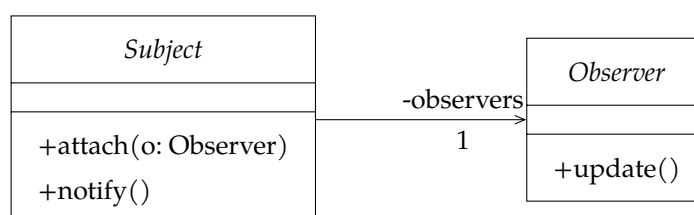
## 66116 / 2018 / Frühjahr / Thema 2 / Teilaufgabe 2 / Aufgabe 2

Es soll eine (kleine) Anwendung entwickelt werden, in der ein Zähler in 1-er Schritten von 5000 bis 0 herunterzählt. Der Zähler soll als Objekt der Klasse `Countdown` realisiert werden, die in UML-Notation dargestellt ist. Das Attribut `value` soll den aktuellen Zählerstand speichern, der mit dem Konstruktor zu initialisieren ist. Die Methode `getValue` soll den aktuellen Zählerstand liefern und die Methode `countdown` soll den Zähler von 5000 bis 0 herunterzählen.

Der jeweilige Zählerstand soll von einem Objekt der in untenstehender Abbildung angegebenen Klasse `Display` am Bildschirm ausgegeben werden. Bei der Konstruktion eines `Display`-Objekts soll es mit einem `Countdown`-Objekt verbunden werden, indem dessen Referenz unter `myCountdown` abgespeichert wird. Die Methode `update` soll den aktuellen Zählerstand vom `Countdown`-Objekt holen und mit `System.out.println` am Bildschirm ausgeben. Dies soll zu Beginn des Zählprozesses und nach jeder Änderung des Zählerstands erfolgen.



Damit das `Display`-Objekt über Zählerstände des `Countdown`-Objekts informiert wird, soll das Observer-Pattern angewendet werden. Untenstehende Abbildung zeigt die im Observer-Pattern vorkommenden abstrakten Klassen. (Kursivschreibweise bedeutet abstrakte Klasse bzw. abstrakte Methode.)



- (a) Welche Wirkung haben die Methoden `attach` und `notify` gemäß der Idee des Observer-Patterns?

Lösungsvorschlag

Das beobachtete Objekt bietet mit der Methode `attach` einen Mechanismus, um Beobachter anzumelden und diese über Änderungen zu informieren.

Mit der Methode `notify` werden alle Beobachter benachrichtigt, wenn sich das beobachtete Objekt ändert.

- (b) Welche der beiden Klassen `Display` und `Countdown` aus obenstehender Abbildung spielt die Rolle eines `Subject` und welche die Rolle eines `Observer`?

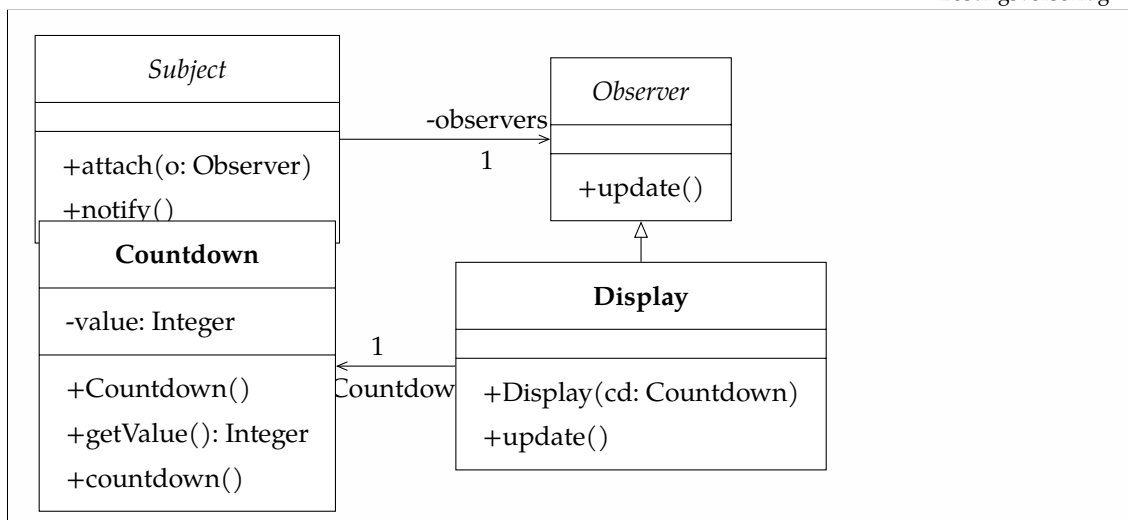
Lösungsvorschlag

Die Klasse `Countdown` spielt die Rolle des `Subject`s, also des Gegenstands, der beobachtet wird.

Die Klasse `Display` spielt die Rolle eines `Observer`, also die Rolle eines Beobachters.

- (c) Erstellen Sie ein Klassendiagramm, das die beiden obenstehenden gegebenen Diagramme in geeigneter Weise, öentsprechend der Idee des Observer-Patterns, zusammenfügt. Es reicht die Klassen und deren Beziehungen anzugeben. Eine nochmalige Nennung der Attribute und Methoden ist nicht notwendig.

Lösungsvorschlag



- (d) Unsere Anwendung soll nun in einer objektorientierten Programmiersprache Ihrer Wahl (z. B. Java oder C++) implementiert werden. Dabei soll von folgenden Annahmen ausgegangen werden:

- Das Programm wird mit einer `main`-Methode gestartet, die folgenden Rumpf hat:

```
public static void main(String[] args){
    Countdown cd = new Countdown();
    new Display(cd);
    cd.countdown();
}
```

- Die beiden Klassen Subject und Observer sind bereits gemäß der Idee des Observer-Patterns implementiert.

Geben Sie auf dieser Grundlage eine Implementierung der beiden Klassen `Display` und `Countdown` an, so dass das gewünschte Verhalten, d.h. Anzeige der Zählerstände und Herunterzählen des Zählers, realisiert wird. Die Methoden der Klassen `Subject` und `Observer` sind dabei auf geeignete Weise zu verwenden bzw. zu implementieren. Geben Sie die verwendete Programmiersprache an.

Lösungsvorschlag

```
public class Client {
    public static void main(String[] args){
        Countdown cd = new Countdown();
        new Display(cd);
        cd.countdown();
        cd.countdown();
        cd.countdown();
    }
}
```

Code-Beispiel auf Github ansehen: [src/main/java/org/bschlangaul/examen/examen\\_66116/jahr\\_2018/fruehjahr/Client.java](https://github.com/bschlangaul/examen/examen_66116/jahr_2018/fruehjahr/Client.java)

```
import java.util.ArrayList;
import java.util.List;

public abstract class Subject {
    private final List<Observer> observers = new ArrayList<Observer>();

    public void attach(Observer o) {
        observers.add(o);
    }

    public void notifyObservers() {
        for (Observer o : observers) {
            o.update();
        }
    }
}
```

Code-Beispiel auf Github ansehen: [src/main/java/org/bschlangaul/examen/examen\\_66116/jahr\\_2018/fruehjahr/Subject.java](https://github.com/bschlangaul/examen/examen_66116/jahr_2018/fruehjahr/Subject.java)

```
public abstract class Observer {
    public abstract void update();
}
```

Code-Beispiel auf Github ansehen: [src/main/java/org/bschlangaul/examen/examen\\_66116/jahr\\_2018/fruehjahr/Observer.java](https://github.com/bschlangaul/examen/examen_66116/jahr_2018/fruehjahr/Observer.java)

```
public class Countdown extends Subject {  
  
    private int value;  
  
    public Countdown() {  
        value = 5000;  
    }  
  
    public int getValue() {  
        return value;  
    }  
  
    public void countdown() {  
        if (value > 0) {  
            notifyObservers();  
            value--;  
        }  
    }  
}
```

Code-Beispiel auf Github ansehen: [src/main/java/org/bschlangaul/examen/examen\\_66116/jahr\\_2018/fruehjahr/Countdown.java](https://github.com/bschlangaul/examen/examen_66116/jahr_2018/fruehjahr/Countdown.java)

```
public class Display extends Observer {  
    Countdown myCountdown;  
    public Display(Countdown cd) {  
        myCountdown = cd;  
        myCountdown.attach(this);  
    }  
  
    public void update() {  
        System.out.println(myCountdown.getValue());  
    }  
}
```

Code-Beispiel auf Github ansehen: [src/main/java/org/bschlangaul/examen/examen\\_66116/jahr\\_2018/fruehjahr/Display.java](https://github.com/bschlangaul/examen/examen_66116/jahr_2018/fruehjahr/Display.java)

## 66116 / 2018 / Herbst / Thema 1 / Teilaufgabe 1 / Aufgabe 2

Gegeben sei das Java-Programm:

```
public class Music {  
    private Beatle state;  
  
    public Music() {  
        state = new Paul();  
    }  
  
    public void help() {  
        this.state.help(this);  
    }  
  
    public void obladi() {  
        this.state.obladi(this);  
    }  
}
```

```
public void yesterday() {
    this.state.yesterday(this);
}

public void setBeatle(Beatle b) {
    state = b;
}
}

abstract class Beatle {
    public abstract void help(Music m);

    public abstract void obladi(Music m);

    public abstract void yesterday(Music m);
}

class George extends Beatle {
    public void help(Music m) {
        System.out.println("help");
        m.setBeatle(new John());
    }

    public void obladi(Music m) {
    }

    public void yesterday(Music m) {
        System.out.println("yesterday");
        m.setBeatle(new Paul());
    }
}

class John extends Beatle {
    public void help(Music m) {
        System.out.println("help");
        m.setBeatle(new Paul());
    }

    public void obladi(Music m) {
        System.out.println("obladi");
        m.setBeatle(new Ringo());
    }

    public void yesterday(Music m) {
    }
}

class Paul extends Beatle {
    public void help(Music m) {
        System.out.println("help");
        m.setBeatle(new George());
    }

    public void obladi(Music m) {
```

```

}

public void yesterday(Music m) {
    System.out.println("yesterday");
}
}

class Ringo extends Beatle {
    public void help(Music m) {
        System.out.println("help");
        m.setBeatle(new John());
    }

    public void obladi(Music m) {
    }

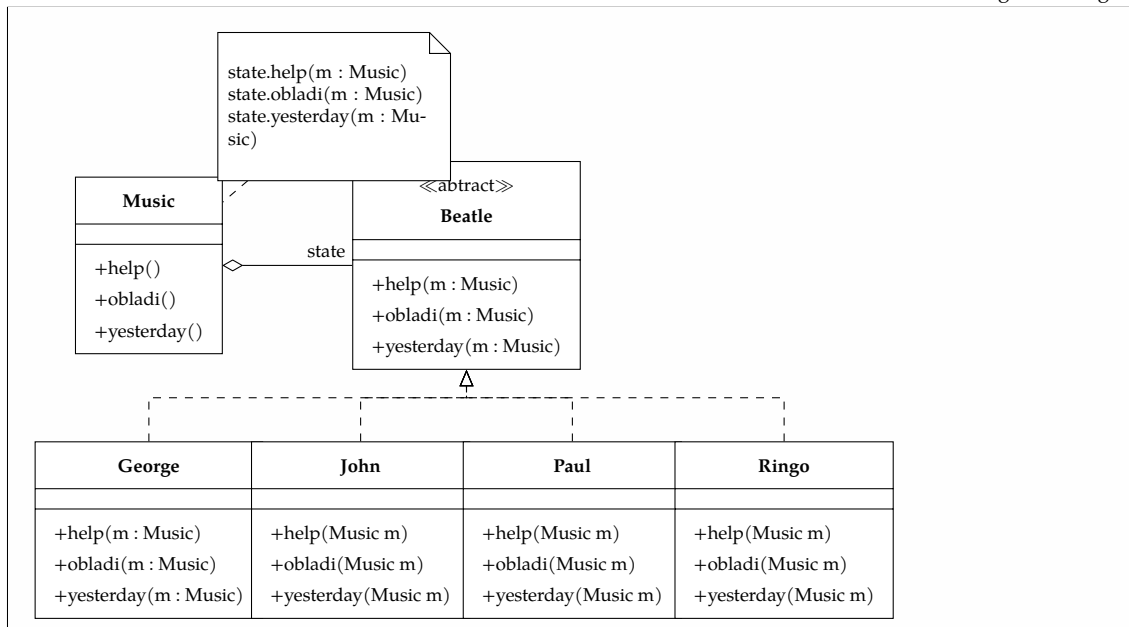
    public void yesterday(Music m) {
        System.out.println("yesterday");
        m.setBeatle(new Paul());
    }
}

```

Code-Beispiel auf Github ansehen: [src/main/java/org/bschlangaul/examen/examen\\_66116/jahr\\_2018/herbst/beatles/Music.java](https://github.com/src/main/java/org/bschlangaul/examen/examen_66116/jahr_2018/herbst/beatles/Music.java)

- (a) Zeichnen Sie ein UML-Klassendiagramm, das die statische Struktur des Programms modelliert. Instanzvariablen mit einem Klassentyp sollen durch gerichtete Assoziationen mit Rollennamen und passender Multiplizität am gerichteten Assoziationsende modelliert werden. Alle aus dem Programmcode ersichtlichen statischen Informationen sollen in dem Klassendiagramm dargestellt werden.

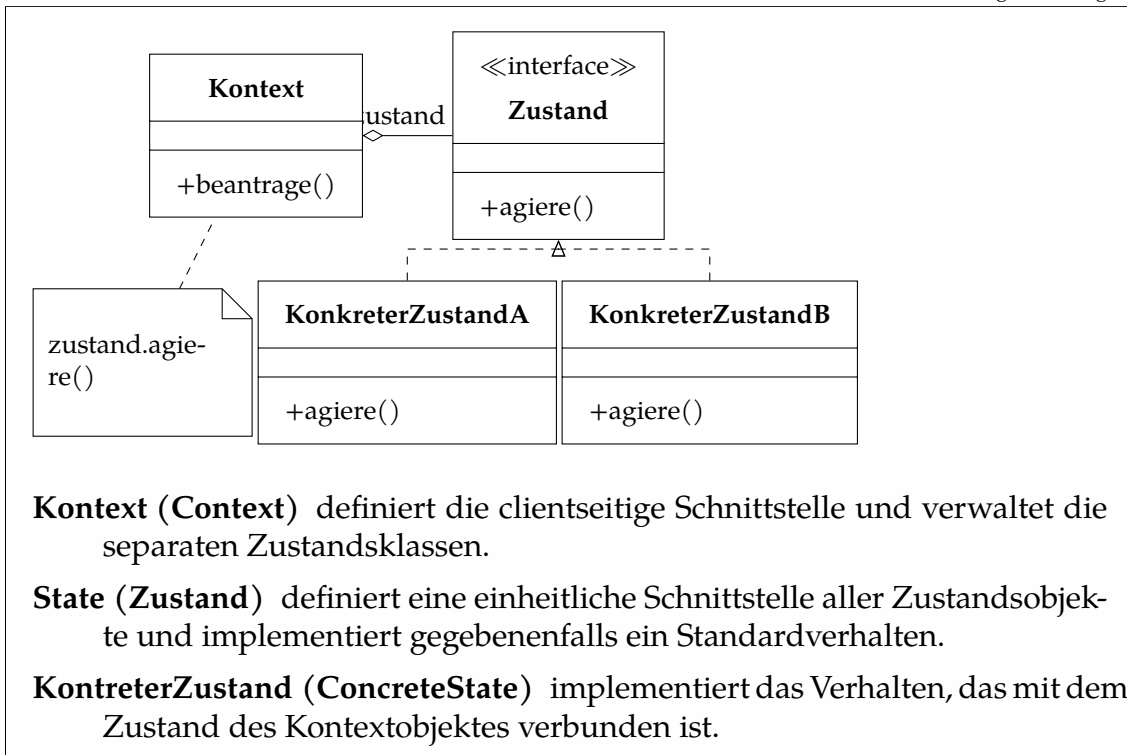
Lösungsvorschlag



Das Programm implementiert ein Zustandsdiagramm, das das Verhalten von Objekten der Klasse `Music` beschreibt. Für die Implementierung wurde das Design-Pattern STATE angewendet.

- (c) Geben Sie die statische Struktur des STATE-Patterns an und erläutern Sie, welche Rollen aus dem Entwurfsmuster den Klassen des gegebenen Programms dabei zufallen und welche Operationen aus dem Entwurfsmuster durch (ggf. mehrere) Methoden in unserem Beispielprogramm implementiert werden. Es ist von den z. B. im Design-Pattern-Katalog von Gamma et al. verwendeten Namen auszugehen, das heißt von Klassen mit Namen `Context`, `State`, `ConcreteStateA`, `ConcreteStateB` und von Operationen mit Namen `request` und `handle`.

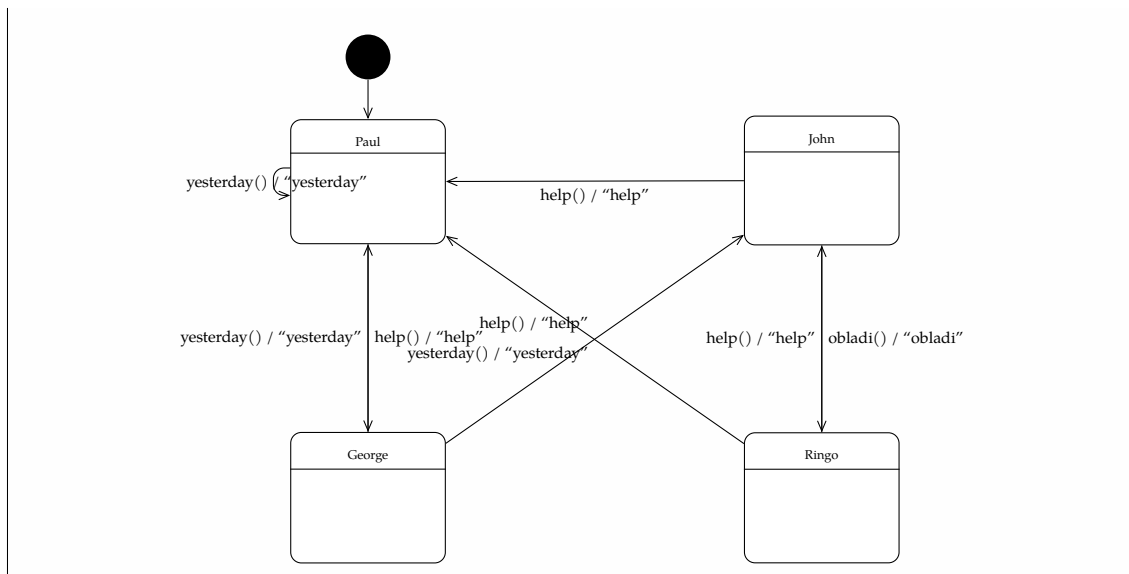
Lösungsvorschlag



- (d) Zeichnen Sie das UML-Zustandsdiagramm (mit Anfangszustand), das von dem Programm implementiert wird. Dabei muss - gemäß der UML-Notation - unterscheidbar sein, was Ereignisse und was Aktionen sind. In dem Diagramm kann zur Vereinfachung statt `System.out.println ("x")` einfach `"x"` geschrieben werden.

Lösungsvorschlag





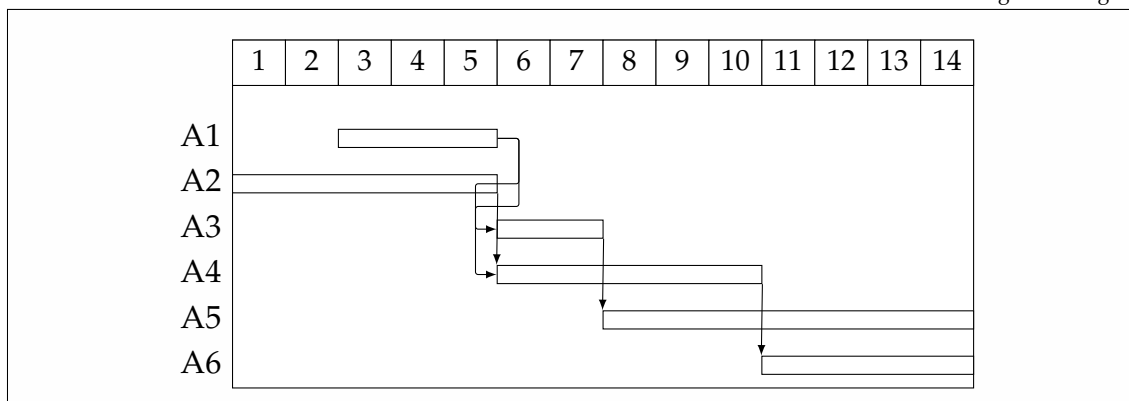
## 66116 / 2018 / Herbst / Thema 2 / Teilaufgabe 1 / Aufgabe 1

Ein Team von zwei Softwareentwicklern soll ein Projekt umsetzen, das in sechs Arbeitspakete unterteilt ist. Die Dauer der Arbeitspakete und ihre Abhängigkeiten können Sie aus folgender Tabelle entnehmen:

Name	Dauer in Wochen	Abhängig von
A1	2	-
A2	5	-
A3	2	A1
A4	5	A1, A2
A5	7	A3
A6	4	A4

- (a) Zeichnen Sie ein Gantt-Diagramm, das eine kürzestmögliche Projektabwicklung beinhaltet.

Lösungsvorschlag





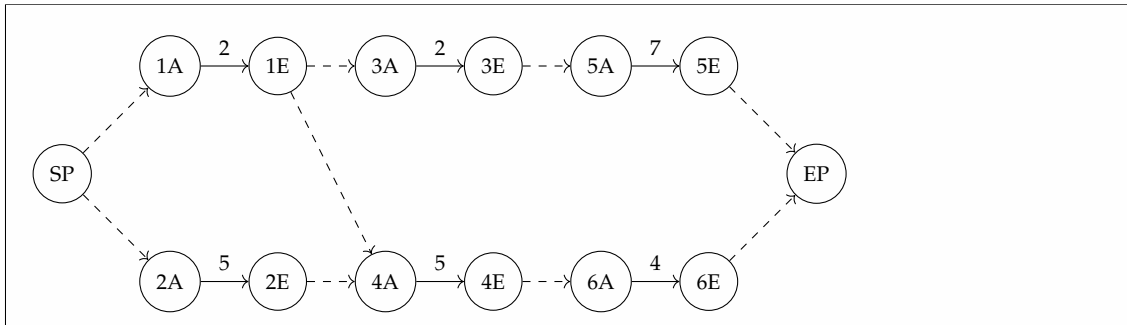
- (b) Bestimmen Sie die Länge des kritischen Pfades und geben Sie an, welche Arbeitspakete an ihm beteiligt sind.

Lösungsvorschlag

Auf dem kritischen Pfad befinden die Arbeitspakete A2, A4 und A6. Die Länge des kritischen Pfades ist 14.

- (c) Wandeln Sie das Gantt-Diagramm in ein CPM-Netzplan um.

Lösungsvorschlag



- (d) Berechnen Sie für jedes Ereignis den *frühesten Termin* und den *spätesten Termin* sowie die *Gesamtpufferzeiten*.

Lösungsvorschlag

<i>i</i>	SP	1A	1E	2A	2E	3A	3E	4A	4E	5A	5E	6A	6E	EP
$FZ_i$	0	0	2	0	5	2	4	5	10	4	11	10	14	14
$SZ_i$	0	3	5	0	5	5	7	5	10	7	14	10	14	14
$GP$	0	3	3	0	0	3	3	0	0	3	3	0	0	0

## 66116 / 2019 / Frühjahr / Thema 1 / Teilaufgabe 2 / Aufgabe 1

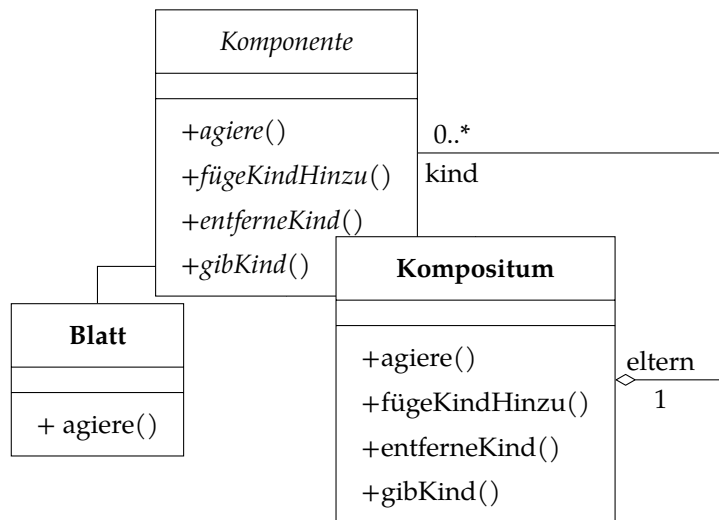
Gegeben sei folgender Sachverhalt: Eine Grafik ist entweder ein Kreis, ein Quadrat oder ein Dreieck. Eine Grafik kann zudem auch eine Kombination aus diesen Elementen sein. Des Weiteren können Sie aus mehreren Grafiken auch neue Grafiken zusammenbauen. Sie denken sich: Ich möchte eine Menge von Grafiken genauso wie eine einzelne Grafik behandeln können.

- (a) Welches Entwurfsmuster sollten Sie zur Modellierung verwenden?

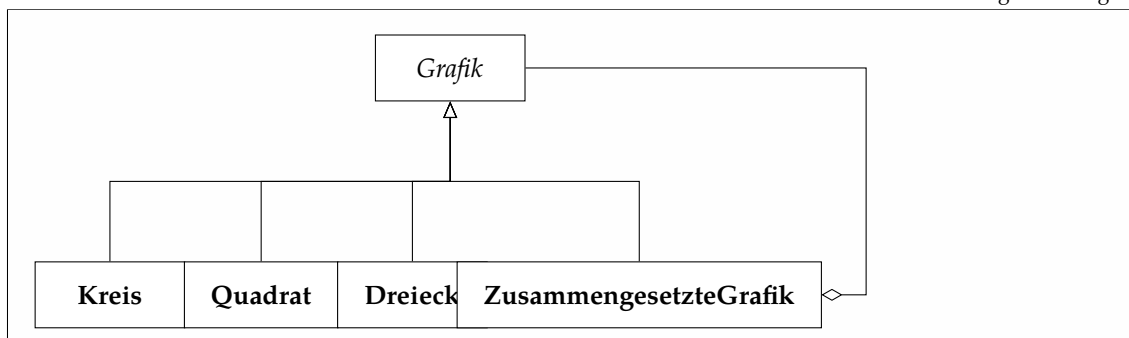
Lösungsvorschlag

Kompositum

- (b) Zeichnen Sie das entsprechende Klassendiagramm. Es reicht, nur die Klassennamen mit ihren Assoziationen und Vererbungsbeziehungen anzugeben; öhne Methoden und Attribute.

**Exkurs: Kompositum**

## Lösungsvorschlag

**66116 / 2019 / Frühjahr / Thema 1 / Teilaufgabe 2 / Aufgabe 2**

Hintergrundinformationen für die Aufgaben 2-5: Modellierung und Implementierung eines Programms

Ein autonomer Roboter in einer Montagehalle bekommt einen Auftrag, eine Menge von Objekten aus dem Lager (Material, z. B. Schrauben oder Werkzeug, z. B. Bohrer) zu holen und anschließend an seinen Ausgangspunkt zu bringen. Der Roboter hat einen Namen, der ihn eindeutig identifiziert, kennt seine aktuelle Position (x- und y-Koordinate) und hat als weitere Eigenschaft den Auftrag, der aus einer Liste von Auftragspositionen besteht, die er holen soll. Der Roboter besitzt folgende Fähigkeiten (Methoden):

- (a) Er kann sich zu einer angegebenen Position bewegen.
- (b) Er hat eine Methode, um einen Auftrag abzuarbeiten, d.h. alle Auftragspositionen zu holen.

Alle Objekte im Lager haben jeweils einen Namen, einen Standort mit Angabe einer Position (s. oben) und speichern außerdem die jeweils noch vorhandene Stückzahl. Es gibt zwei Typen von Objekten: Werkzeuge mit einer Methode, mit der ein einzelnes Werkzeug ausgeliehen werden kann, und Materialien mit einer Methode, mit der sie verbraucht werden, wobei eine gewünschte Stückzahl angegeben wird. Diese vorgegebenen Methoden aktualisieren die Stückzahlen der Werkzeuge (Reduktion um 1) bzw. Materialien (Reduktion um verbrauchte Stückzahl), wobei hier vereinfachend angenommen wird, dass die Stückzahlen der Werkzeuge und Materialien immer ausreichend

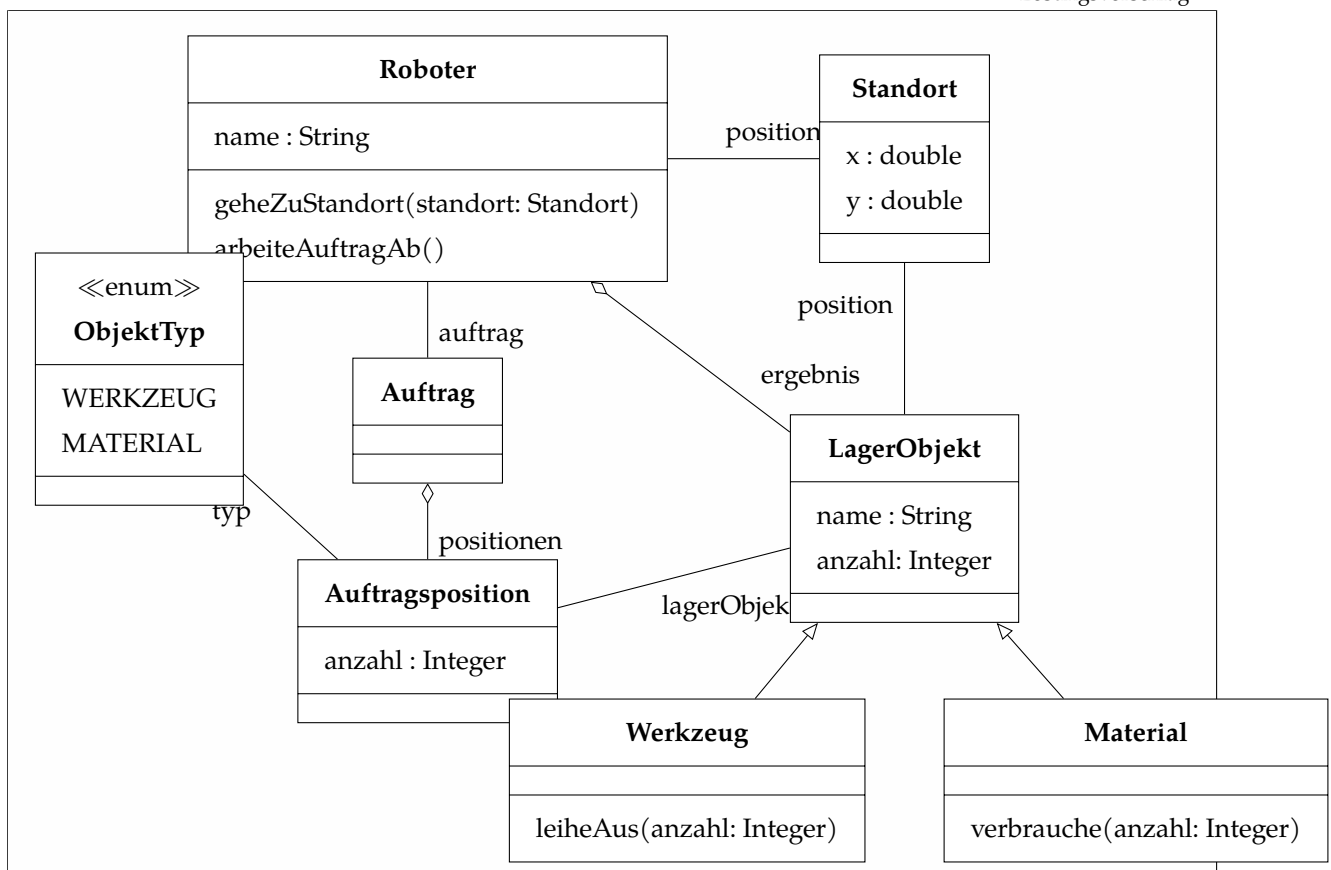
groß sind, um die geforderten Mengen bedienen zu können (d.h. Sie können diese beiden Methoden nutzen, ohne deren Implementierung angeben zu müssen).

Ein Auftrag (z. B. „Hole Bohrer Typ B1, 100 × Schrauben M6, 10 × Schrauben M10 und 2 × Blech B72“) besteht aus einer Menge von Auftragspositionen. Eine Auftragsposition besteht aus dem Typ des zu holenden Objekts (Werkzeug oder Material, soll als Enumeration modelliert werden), einem Verweis auf das zu holende Objekt und der zu holenden Stückzahl (Quantität; bei Werkzeugen wird diese ignoriert, da sie immer 1 ist). Der Roboter soll über die Auftragsposition außerdem die Position bestimmen, zu der er fahren muss, um das Objekt zu holen.

Der Roboter arbeitet die Auftragspositionen in der Reihenfolge ab, indem er sich von seinem aktuellen Standpunkt immer zur am nächsten liegenden Auftragsposition bewegt, um dort das nächste Objekt zu holen. Wir gehen der Einfachheit halber davon aus, dass die Montagehalle gut aufgeräumt ist und der Roboter sich quasi entlang der Luftlinie bewegen kann, d.h. die Entfernung zwischen zwei Positionen entspricht der euklidischen Distanz (Wurzel aus der Summe der Quadrate der Differenzen der x- und y-Koordinaten der Positionen). Der Roboter soll zur Kontrolle als weitere Eigenschaft „Ergebnis“ die Liste der eingesammelten Objekte zu einem Auftrag in der Reihenfolge speichern, in der er sie geholt hat, z.B. (M6 M10 B72 B1).

Geben Sie ein UML-Klassendiagramm zu der Aufgabenstellung an. Hinweis: Bei den Aufgaben 4 und 5 wird Konsistenz des Aktivitätsdiagramms bzw. des Codes mit dem Klassendiagramm verlangt.

Lösungsvorschlag



## 66116 / 2019 / Frühjahr / Thema 1 / Teilaufgabe 2 / Aufgabe 3

- (a) Erläutern Sie kurz, was man unter der Methode der testgetriebenen Entwicklung versteht.

Bei der testgetriebenen Entwicklung erstellt der/die ProgrammiererIn Softwaretests konsequent vor den zu testenden Komponenten.

- (b) Geben Sie für obige Aufgabenstellung (Abarbeitung der Aufträge durch den Roboter) einen Testfall für eine typische Situation an (d. h. das Einsammeln von 4 Objekten an unterschiedlichen Positionen). Spezifizieren Sie als Input alle für die Abarbeitung eines Auftrages relevanten Eingabe- und Klassen-Parameter sowie den vollständigen und korrekten Output.

### 66116 / 2019 / Frühjahr / Thema 1 / Teilaufgabe 2 / Aufgabe 4

- (a) Geben Sie für alle Methoden, die zur Abarbeitung des Auftrages für den Roboter erforderlich sind, Pseudocode an. Nutzen Sie (im Klassendiagramm definierte) Hilfsmethoden, um den Code übersichtlicher zu gestalten. (Verwenden Sie statt einer komplizierten Methode mehrere einfachere Methoden, die sich aufrufen.)  
Achten Sie auf die Konsistenz zum Klassendiagramm (Inkonsistenzen führen zu Punktabzügen).
- (b) Überführen Sie den Pseudocode der Hauptmethode zur Abarbeitung eines Auftrags in ein syntaktisch korrektes UML-Aktivitätsdiagramm. Schleifen und Verzweigungen müssen durch strukturierte Knoten dargestellt werden (d. h. kein graphisches "goto").

### 66116 / 2019 / Frühjahr / Thema 2 / Teilaufgabe 2 / Aufgabe 1

Bewerten Sie die folgenden Aussagen und nehmen Sie jeweils Stellung.

- (a) Tests zuerst

In test-getriebener Softwareentwicklung wird der Test vor der zu testenden Funktion programmiert.

Bei der testgetriebenen Entwicklung erstellt der/die ProgrammiererIn Softwaretests konsequent vor den zu testenden Komponenten.

- (b) Komponententests

Komponententests sind immer White-Box-Tests und nie Black-Box-Tests.

Falsch: Komponententests (anderes Wort für Unit- oder Modul-Tests) können beides sein. <sup>a</sup>

<sup>a</sup><https://qastack.com.de/software/362746/what-is-black-box-unit-testing>

(c) Akzeptanztests

Akzeptanz - resp. Funktionstests sind immer Black-Box-Tests und nie White-Box-Tests.

Lösungsvorschlag

In systems engineering, it may involve black-box testing performed on a system

Acceptance testing in extreme programming: Acceptance tests are black-box system tests.

<sup>a</sup>

<sup>a</sup>[https://en.wikipedia.org/wiki/Acceptance\\_testing](https://en.wikipedia.org/wiki/Acceptance_testing)

(d) Fehler

Ein fehlgeschlagener Test und ein Testausführungsfehler bezeichnen denselben Sachverhalt.

Lösungsvorschlag

Falsch:

Fehler (Error)

Der Software-Test konnte durchgeführt werden, das ermittelte Ist-Ergebnis und das vorgegebene Soll-Ergebnis weichen jedoch voneinander ab. Ein derartiger Fehler liegt z. B. dann vor, wenn ein Funktionsaufruf einen abweichenden Wert berechnet.

Fehlschlag (Failure)

Während der Testdurchführung wurde ein Systemversagen festgestellt. Ein Fehlschlag liegt z. B. dann vor, wenn das zu testende System einen Programmabsturz verursacht und hierdurch erst gar kein Ist-Ergebnis ermittelt werden konnte.

Das Misslingen kann als Ursache einen Fehler (Error) oder ein falsches Ergebnis (Failure) haben, die beide per Exception signalisiert werden. Der Unterschied zwischen den beiden Begriffen liegt darin, dass Failures erwartet werden, während Errors eher unerwartet auftreten. <sup>a</sup>

<sup>a</sup><https://de.wikipedia.org/wiki/JUnit>

(e) Test Suiten

Tests können hierarchisch strukturiert werden, so dass mit einem Befehl das gesamte zu testende System getestet werden kann.

Lösungsvorschlag

Richtig:

test suite (englisch „Testsammlung“, aus französisch suite Folge, Verkettung) bezeichnet eine organisierte Sammlung von Werkzeugen zum Testen techni-

scher Apparate und Vorgänge.<sup>a</sup>

<sup>a</sup><https://de.wikipedia.org/wiki/Testsuite>

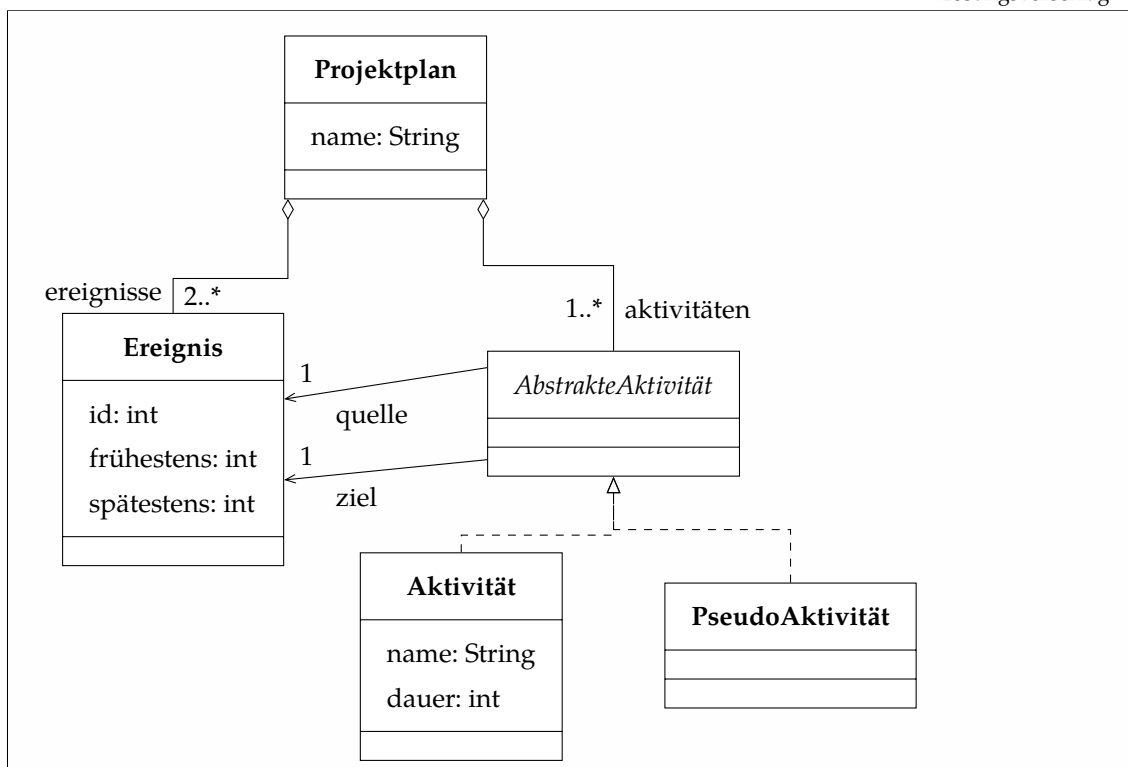
Klassendiagramm  
Objektdiagramm

## 66116 / 2019 / Herbst / Thema 1 / Teilaufgabe 1 / Aufgabe 1

Ein CPM-Netzwerk („Critical Path Method“) ist ein benannter Projektplan, der aus Ereignissen und Aktivitäten besteht. Ein Ereignis wird durch eine ganze Zahl  $> 0$  identifiziert. Jede Aktivität führt von einem Quellereignis zu einem Zielereignis. Eine reale Aktivität hat einen Namen und eine Dauer (eine ganze Zahl  $> 0$ ). Eine Pseudoaktivität ist anonym. Ereignisse und Pseudoaktivitäten verbrauchen keine Zeit. Zu jedem Ereignis gibt es einen frühesten und einen spätesten Zeitpunkt (eine ganze Zahl  $> 0$ ), deren Berechnung nicht Gegenstand der Aufgabe ist.

- (a) Erstellen Sie ein UML-Klassendiagramm zur Modellierung von CPM-Netzwerken. Geben Sie für Attribute jeweils den Namen und den Typ an. Geben Sie für Assoziationen den Namen und für jedes Ende den Rollennamen und die Multiplizität an. Nutzen Sie ggf. abstrakte Klassen, Vererbung, Komposition oder Aggregation. Verzichten Sie auf Operationen und Sichtbarkeiten.

Lösungsvorschlag



- (b) Erstellen Sie für das Klassendiagramm aus a) und das Beispiel aus der Aufgabenstellung ein Objektdiagramm. Geben Sie Rollennamen nur an, wenn es notwendig ist, um die Enden eines Links (Instanz einer Assoziation) zu unterscheiden.

**66116 / 2019 / Herbst / Thema 1 / Teilaufgabe 1 / Aufgabe 2**

Eine Dienstreise an einer Universität wird folgendermaßen abgewickelt:

Ein Mitarbeiter erstellt einen Dienstreiseantrag und legt ihn seinem Vorgesetzten zur Unterschrift vor. Mit seiner Unterschrift befürwortet der Vorgesetzte die Dienstreise. Verweigert er die Unterschrift, wird der Vorgang abgebrochen. Nach der Befürwortung wird der Antrag an die Reisekostenstelle weitergeleitet, die über die Annahme des Antrags entscheidet. Im Falle einer Ablehnung wird der Vorgang abgebrochen; sonst genehmigt die Reisekostenstelle den Antrag. Der Mitarbeiter kann nun einen Abschlag beantragen. Stimmt der Vorgesetzte zu, so entscheidet die Reisekostenstelle über den Abschlag und weist ggf. die Kasse der Universität an, den Abschlag auszuzahlen. Nach Ende der Dienstreise erstellt der Mitarbeiter einen Antrag auf Erstattung der Reisekosten an die Reisekostenstelle. Die Reisekostenstelle setzt den Erstattungsbetrag fest und weist die Kasse an, den Betrag auszuzahlen.

- Erstellen Sie ein Anwendungsfalldiagramm (Use-Case-Diagramm) für Dienstreisen.
- Erstellen Sie ein Aktivitätsdiagramm, das den oben beschriebenen Ablauf modelliert. Ordnen Sie den Aktionen die Akteure gemäß a) zu. Beschränken Sie sich auf den Kontrollfluss (keine Objektflüsse).

**66116 / 2019 / Herbst / Thema 1 / Teilaufgabe 1 / Aufgabe 3**

Eine Dezimalzahl hat ein optionales Vorzeichen, dem eine nichtleere Sequenz von Dezimalziffern folgt. Der anschließende gebrochene Anteil ist optional und besteht aus einem Dezimalpunkt, gefolgt von einer nichtleeren Sequenz von Dezimalziffern.

Die folgende Java-Methode erkennt, ob eine Zeichenfolge eine Dezimalzahl ist:

```
public static boolean istDezimalzahl(char[] zeichen) { // 1
    boolean resultat; int laenge = zeichen.length; // 2
    if (laenge == 0) // 3
        resultat = false; // 4
    else { // 5
        int i = 0; // 6
        if (zeichen[i] == '+' || zeichen[i] == '-') // 7
            i++; // 8
        int j = i; // 9
        while (i < laenge && '0' <= zeichen[i] && zeichen[i] <= '9') // 10
            i++; // 11
        if (i == j) // 12
            resultat = false; // 13
        else { // 14
            if (i < laenge && zeichen[i] == '.') // 15
                do // 16
                    i++; // 17
                while (i < laenge && '0' <= zeichen[i] && zeichen[i] <= '9'); // 18
            resultat = i == laenge && '0' <= zeichen[i - 1] && zeichen[i - 1] <= '9';
        } // 19
    } // 20
} // 21
```

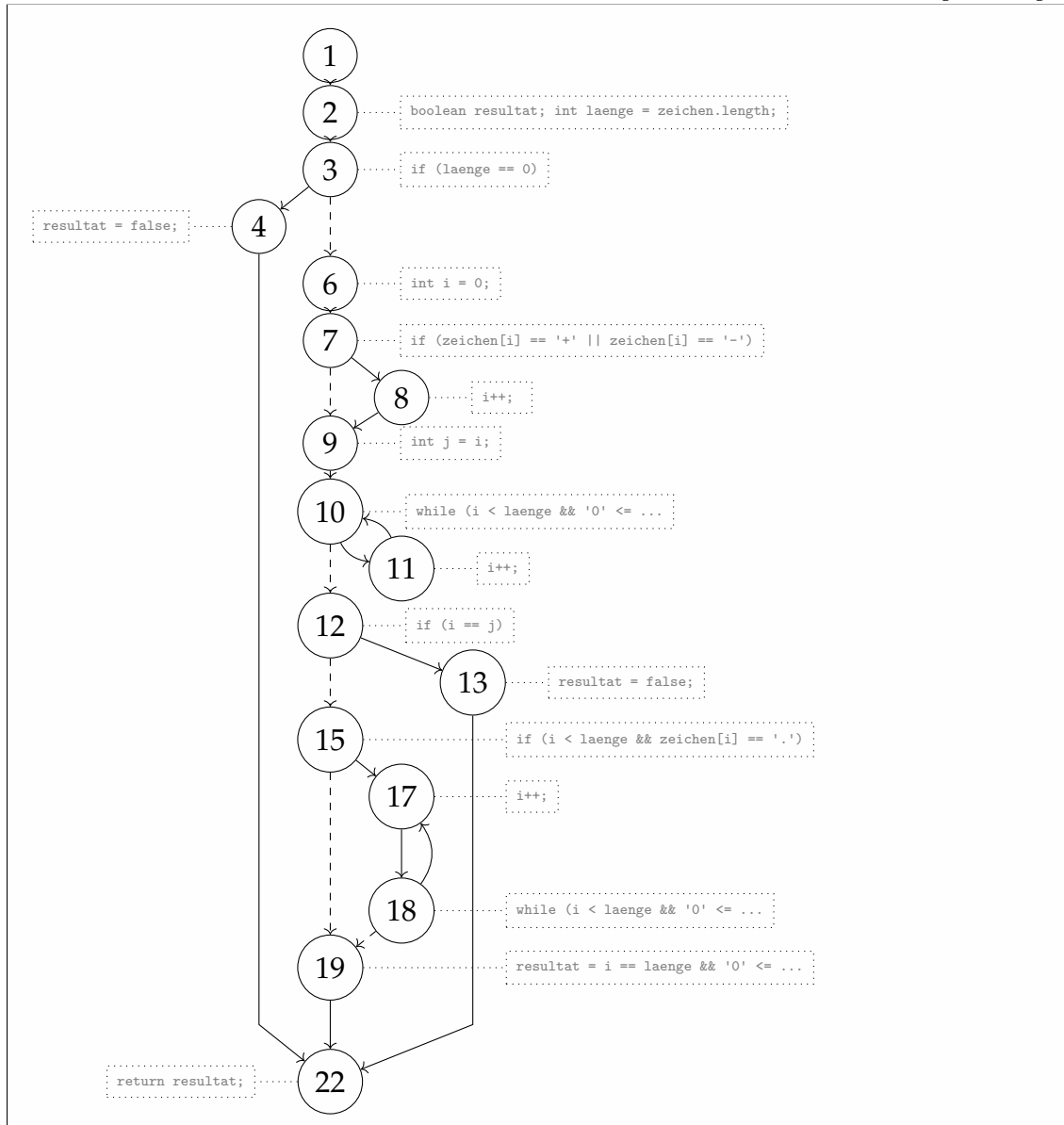
```

return resultat; // 22
}

```

- (a) Konstruieren Sie zu dieser Methode einen Kontrollflußgraphen. Markieren Sie dessen Knoten mit Zeilennummern des Quelltexts.

Lösungsvorschlag



- (b) Geben Sie eine minimale Testmenge an, die das Kriterium der Knotenüberdeckung erfüllt. Geben Sie für jeden Testfall den durchlaufenen Pfad in der Notation  $1 \rightarrow 2 \rightarrow \dots$  an.

Lösungsvorschlag

```

- "0":
  1 - 2 - 3 - 4 - 22
- "1":
  1 - 2 - 3 - 6 - 7 - 9 - 10 - 11 - 10 - 12 - 15 - 19 - 22

```



- „+1.0“:

① - ② - ③ - ⑥ - ⑦ - ⑧ - ⑨ - ⑩ - ⑪ - ⑩ - ⑫ - ⑮ - ⑰ - ⑱ - ⑳

- „X“:

① - ② - ③ - ⑥ - ⑦ - ⑨ - ⑩ - ⑫ - ⑬ - ⑳

Zustandsdiagramm Wissen  
Zustand (State)

(c) Verfahren Sie wie in b) für das Kriterium der Kantenüberdeckung.

Lösungsvorschlag

text

(d) Wie stehen die Kriterien der Knoten- und Kantenüberdeckung zueinander in Beziehung? Begründen Sie Ihre Antwort.

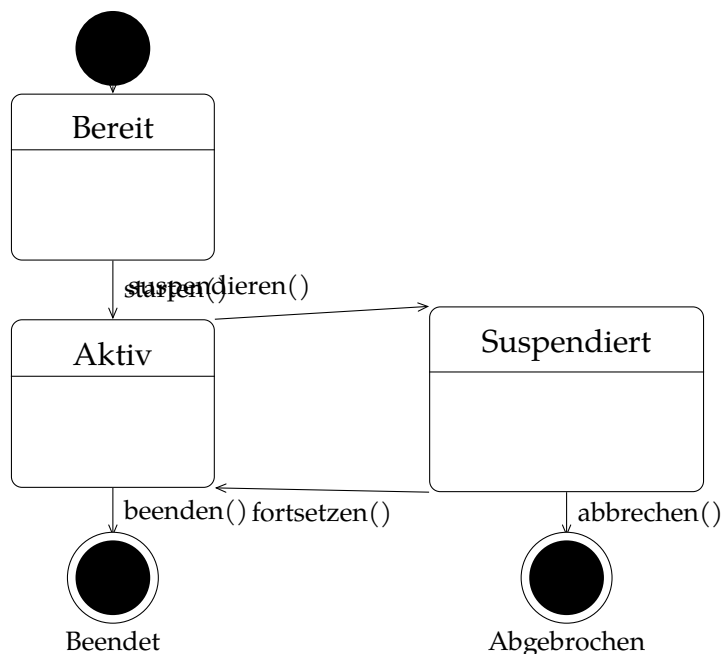
Hinweis: Eine Testmenge ist minimal, wenn es keine andere Testmenge mit einer kleineren Zahl von Testfällen gibt. Die Minimalität braucht nicht bewiesen zu werden.

Lösungsvorschlag

Die Kantenüberdeckung fordert, dass jede *Kante* des Kontrollflussgraphen von mindestens einem Testfall durchlaufen werden muss. Um das Kriterium zu erfüllen, müssen die Testfälle so gewählt werden, dass jede Verzweigungsbedingung mindestens *einmal wahr* und mindestens *einmal falsch* wird. Da hierdurch alle Knoten ebenfalls mindestens einmal besucht werden müssen, ist die *Anweisungsüberdeckung* in der Zweigüberdeckung *vollständig enthalten*.

## 66116 / 2019 / Herbst / Thema 1 / Teilaufgabe 1 / Aufgabe 4

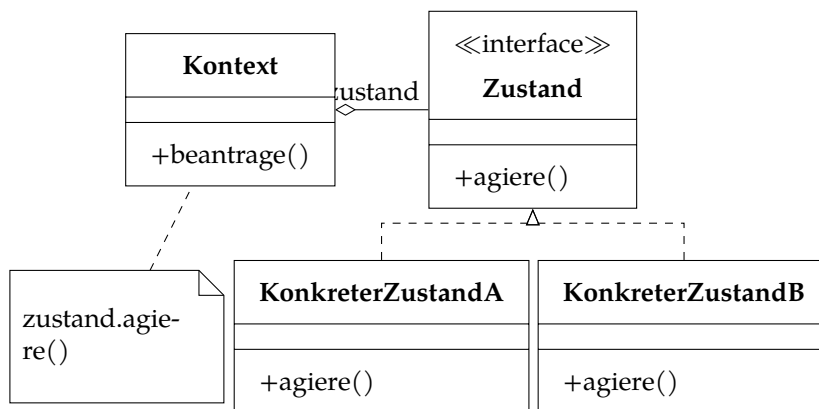
Zu den Aufgaben eines Betriebssystems zählt die Verwaltung von Prozessen. Jeder Prozess durchläuft verschiedene Zustände; Transitionen werden durch Operationsaufrufe ausgelöst. Folgendes Zustandsdiagramm beschreibt die Verwaltung von Prozessen:



Implementieren Sie dieses Zustandsdiagramm in einer Programmiersprache Ihrer Wahl mit Hilfe des Zustandsmusters; geben Sie die gewählte Sprache an. Die Methoden für die Transitionen sollen dabei die Funktionalität der Prozessverwaltung simulieren, indem der Methodenaufruf auf der Standardausgabe protokolliert wird. Falls Transitionen im aktuellen Zustand undefiniert sind, soll eine Fehlermeldung ausgegeben werden.

### Exkurs: Zustand-(State)-Entwurfsmuster

#### UML-Klassendiagramm



#### Teilnehmer

**Kontext (Context)** definiert die clientseitige Schnittstelle und verwaltet die separaten Zustandsklassen.

**State (Zustand)** definiert eine einheitliche Schnittstelle aller Zustandsobjekte und implementiert gegebenenfalls ein Standardverhalten.

**KonkreterZustand (ConcreteState)** implementiert das Verhalten, das mit dem Zustand des Kontextobjektes verbunden ist.

Implementierung in der Programmiersprache „Java“:

Methoden	Zustände	Klassennamen
	Bereit	ZustandBereit
starten(), fortsetzen()	Aktiv	ZustandAktiv
suspendieren()	Suspendiert	ZustandSuspendiert
beenden()	Beendet	ZustandBeendet
abbrechen()	Abgebrochen	ZustandAbgebrochen

```
/**  
 * Entspricht der „Kontext“-Klasse in der Terminologie der „Gang of  
 * Four“.
```

```
*/
public class Prozess {

    private ProzessZustand aktuellerZustand;

    public Prozess() {
        aktuellerZustand = new ZustandBereit(this);
    }

    public void setzeZustand(ProzessZustand zustand) {
        aktuellerZustand = zustand;
    }

    public void starten() {
        aktuellerZustand.starten();
    }

    public void suspendieren() {
        aktuellerZustand.suspendieren();
    }

    public void fortsetzen() {
        aktuellerZustand.fortsetzen();
    }

    public void beenden() {
        aktuellerZustand.beenden();
    }

    public void abbrechen() {
        aktuellerZustand.abbrechen();
    }

    public static void main(String[] args) {
        Prozess prozess = new Prozess();
        prozess.starten();
        prozess.suspendieren();
        prozess.fortsetzen();
        prozess.beenden();
        prozess.starten();

        // Ausgabe:
        // Der Prozess ist im Zustand „bereit“
        // Der Prozess wird gestartet.
        // Der Prozess ist im Zustand „aktiv“
        // Der Prozess wird suspendiert.
        // Der Prozess ist im Zustand „suspendiert“
        // Der Prozess wird fortgesetzt.
        // Der Prozess ist im Zustand „aktiv“
        // Der Prozess wird beendet.
        // Der Prozess ist im Zustand „beendet“
        // Im Zustand „beendet“ kann die Transition „starten“ nicht ausgeführt werden!
    }
}
```

Code-Beispiel auf Github ansehen: [src/main/java/org/bschlangaul/examen/examen\\_66116/jahr\\_2019/herbst/prozess\\_verwaltung/Prozess.java](https://github.com/bschlangaul/examen/examen_66116/jahr_2019/herbst/prozess_verwaltung/Prozess.java)

```
/**
 * Entspricht der „Zustand“-Klasse in der Terminologie der "Gang of
 * Four".
 */
abstract class ProzessZustand {

    Prozess prozess;

    String zustand;

    public ProzessZustand(String zustand, Prozess prozess) {
        this.zustand = zustand;
        this.prozess = prozess;
        System.out.println(String.format("Der Prozess ist im Zustand „%s“, zustand));
    }

    private void gibFehlermeldungAus(String transition) {
        System.err.println(
→ String.format("Im Zustand „%s“ kann die Transition „%s“ nicht ausgeführt werden!",
        zustand, transition));
    }

    public void starten() {
        gibFehlermeldungAus("starten");
    }

    public void suspendieren() {
        gibFehlermeldungAus("suspendieren");
    }

    public void fortsetzen() {
        gibFehlermeldungAus("fortsetzen");
    }

    public void beenden() {
        gibFehlermeldungAus("beenden");
    }

    public void abbrechen() {
        gibFehlermeldungAus("abbrechen");
    }
}
```

Code-Beispiel auf Github ansehen: [src/main/java/org/bschlangaul/examen/examen\\_66116/jahr\\_2019/herbst/prozess\\_verwaltung/ProzessZustand.java](https://github.com/bschlangaul/examen/examen_66116/jahr_2019/herbst/prozess_verwaltung/ProzessZustand.java)

```
/**
 * Entspricht der „KonkreterZustand“-Unterklasse in der Terminologie der "Gang of
 * Four".
 */
public class ZustandAbgebrochen extends ProzessZustand {
```

```

    public ZustandAbgebrochen(Prozess prozess) {
        super("abgebrochen", prozess);
    }
}

```

Code-Beispiel auf Github ansehen: [src/main/java/org/bschlangaul/examen/examen\\_66116/jahr\\_2019/herbst/prozess\\_verwaltung/ZustandAbgebrochen.java](https://github.com/bschlangaul/examen_exam_66116_jahr_2019_herbst_prozess_verwaltung_ZustandAbgebrochen.java)

```

/**
 * Entspricht der „KonkreterZustand“-Unterklasse in der Terminologie der „Gang of
 * Four“.
 */
public class ZustandAktiv extends ProzessZustand {

    public ZustandAktiv(Prozess prozess) {
        super("aktiv", prozess);
    }

    public void suspendieren() {
        System.out.println("Der Prozess wird suspendiert.");
        prozess.setzeZustand(new ZustandSuspendiert(prozess));
    }

    public void beenden() {
        System.out.println("Der Prozess wird beendet.");
        prozess.setzeZustand(new ZustandBeendet(prozess));
    }
}

```

Code-Beispiel auf Github ansehen: [src/main/java/org/bschlangaul/examen/examen\\_66116/jahr\\_2019/herbst/prozess\\_verwaltung/ZustandAktiv.java](https://github.com/bschlangaul/examen_exam_66116_jahr_2019_herbst_prozess_verwaltung_ZustandAktiv.java)

```

/**
 * Entspricht der „KonkreterZustand“-Unterklasse in der Terminologie der „Gang of
 * Four“.
 */
public class ZustandBeendet extends ProzessZustand {

    public ZustandBeendet(Prozess prozess) {
        super("beendet", prozess);
    }
}

```

Code-Beispiel auf Github ansehen: [src/main/java/org/bschlangaul/examen/examen\\_66116/jahr\\_2019/herbst/prozess\\_verwaltung/ZustandBeendet.java](https://github.com/bschlangaul/examen_exam_66116_jahr_2019_herbst_prozess_verwaltung_ZustandBeendet.java)

```

/**
 * Entspricht der „KonkreterZustand“-Unterklasse in der Terminologie der „Gang of
 * Four“.
 */
public class ZustandBereit extends ProzessZustand {

    public ZustandBereit(Prozess prozess) {
        super("bereit", prozess);
    }

    public void starten() {

```

```

        System.out.println("Der Prozess wird gestartet.");
        prozess.setzeZustand(new ZustandAktiv(prozess));
    }
}

```

Code-Beispiel auf Github ansehen: [src/main/java/org/bschlangaul/examen/examen\\_66116/jahr\\_2019/herbst/prozess\\_verwaltung/ZustandBereit.java](https://github.com/bschlangaul/examen/examen_66116/jahr_2019/herbst/prozess_verwaltung/ZustandBereit.java)

```

/**
 * Entspricht der „KonkreterZustand“-Unterklasse in der Terminologie der „Gang of
 * Four“.
 */
public class ZustandSuspendiert extends ProzessZustand {

    public ZustandSuspendiert(Prozess prozess) {
        super("suspendiert", prozess);
    }

    public void fortsetzen() {
        System.out.println("Der Prozess wird fortgesetzt.");
        prozess.setzeZustand(new ZustandAktiv(prozess));
    }

    public void abbrechen() {
        System.out.println("Der Prozess wird abgebrochen.");
        prozess.setzeZustand(new ZustandAbgebrochen(prozess));
    }
}

```

Code-Beispiel auf Github ansehen:

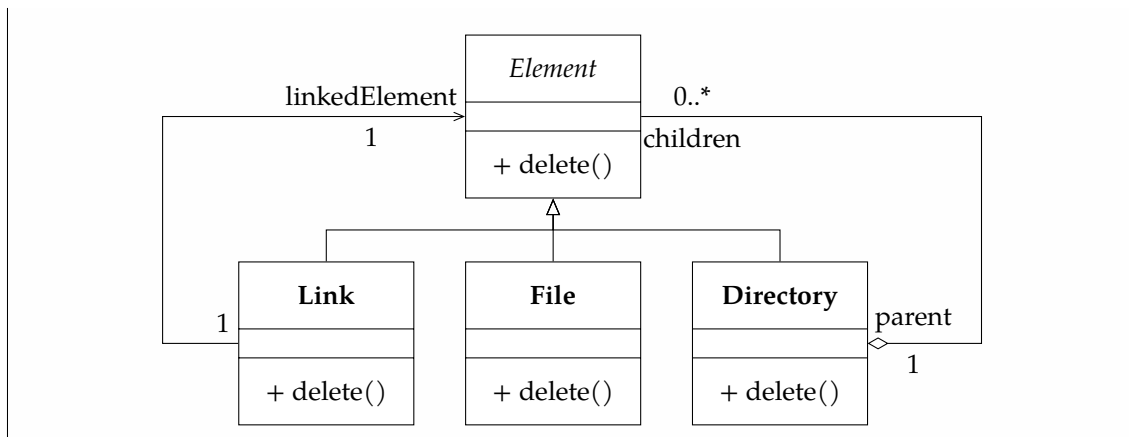
[src/main/java/org/bschlangaul/examen/examen\\_66116/jahr\\_2019/herbst/prozess\\_verwaltung/ZustandSuspendiert.java](https://github.com/bschlangaul/examen/examen_66116/jahr_2019/herbst/prozess_verwaltung/ZustandSuspendiert.java)

## 66116 / 2019 / Herbst / Thema 2 / Teilaufgabe 1 / Aufgabe 1

Hierarchische Dateisysteme bestehen aus den FileSystemElements Ordner, Dateien und Verweise. Ein Ordner kann seinerseits Ordner, Dateien und Verweise beinhalten; jedem Ordner ist bekannt, welche Elemente (children) er enthält. Mit Ausnahme des Root-Ordners auf der obersten Hierarchieebene ist jeder Ordner, jede Datei und jeder Verweis Element eines Elternordners. Jedem Element ist bekannt, was sein Elternordner ist (parent). Ein Verweis verweist auf einen Verweis, eine Datei oder einen Ordner (link). Wenn ein Ordner gelöscht wird, werden alle seine Bestandteile ggf. rekursiv ebenfalls gelöscht. Sie dürfen die Lösungen für Aufgabenteil b) und c) in einem gemeinsamen Code kombinieren.

- (a) Modellieren Sie diesen Sachverhalt mit einem UML-Klassendiagramm. Benennen Sie die Rollen von Assoziationen und geben Sie alle Kardinalitäten an. Ihre Lösung soll mindestens eine sinnvolle Spezialisierungsbeziehung enthalten.

Lösungsvorschlag



- (b) Implementieren Sie das Klassendiagramm als Java- oder C++-Programm. Jedes Element des Dateisystems soll mindestens über ein Attribut `name` verfügen. Übergeben Sie den Elternordner jedes Elements als Parameter an den Konstruktor; der Elternordner des Root-Ordners kann dabei als `null` implementiert werden. Dokumentieren Sie Ihren Code.



*a*

```
public abstract class Element {  
  
    protected String name;  
  
    protected Element parent;  
  
    protected Element(String name, Element parent) {  
        this.name = name;  
        this.parent = parent;  
    }  
  
    public String getName() {  
        return name;  
    }  
  
    public abstract void delete();  
  
    public abstract boolean isDirectory();  
  
    public abstract void addChild(Element child);  
}
```

Code-Beispiel auf Github ansehen: [src/main/java/org/bschlangaul/examen/examen\\_66116/jahr\\_2019/herbst/file\\_system/Element.java](https://github.com/bschlangaul/examen/examen_66116/jahr_2019/herbst/file_system/Element.java)

```
import java.util.ArrayList;  
import java.util.List;  
  
public class Directory extends Element {  
    private List<Element> children;  
  
    public Directory(String name, Element parent) {
```

```

    super(name, parent);
    children = new ArrayList<Element>();
    if (parent != null)
        parent.addChild(this);
}

public void delete() {
    System.out.println("The directory " + name +
→    " was deleted and it's children were also deleted.");
    for (int i = 0; i < children.size(); i++) {
        Element child = children.get(i);
        child.delete();
    }
}

public void addChild(Element child) {
    children.add(child);
}

public boolean isDirectory() {
    return true;
}
}

```

Code-Beispiel auf Github ansehen: [src/main/java/org/bschlangaul/examen/examen\\_66116/jahr\\_2019/herbst/file\\_system/Directory.java](https://github.com/bschlangaul/examen/examen_66116/jahr_2019/herbst/file_system/Directory.java)

```

public class File extends Element {

    public File(String name, Element parent) {
        super(name, parent);
        parent.addChild(this);
    }

    public void delete() {
        System.out.println("The File " + name + " was deleted.");
    }

    public boolean isDirectory() {
        return false;
    }

    /**
     * Eine Datei kann keine Kinder haben. Deshalb eine Methode mit leerem
     * Methodenrumpf.
     */
    public void addChild(Element child) {
    }

}

```

Code-Beispiel auf Github ansehen: [src/main/java/org/bschlangaul/examen/examen\\_66116/jahr\\_2019/herbst/file\\_system/File.java](https://github.com/bschlangaul/examen/examen_66116/jahr_2019/herbst/file_system/File.java)

```

public class Link extends Element {

```

```

private Element linkedElement;

public Link(String name, Element parent, Element linkedElement) {
    super(name, parent);
    this.linkedElement = linkedElement;
    parent.addChild(this);
}

public void delete() {
    System.out.println("The Symbolic Link " + name + " was deleted.");
    linkedElement.delete();
    System.out.println("The linked element " + name + " was deleted too.");
}

public void addChild(Element child) {
    if (linkedElement.isDirectory())
        linkedElement.addChild(child);
}

public boolean isDirectory() {
    return linkedElement.isDirectory();
}
}

```

Code-Beispiel auf Github ansehen: [src/main/java/org/bschlangaul/examen/examen\\_66116/jahr\\_2019/herbst/file\\_system/Link.java](https://github.com/bschlangaul/examen/examen_66116/jahr_2019/herbst/file_system/Link.java)

"TU Darmstadt: Dr. Michael Eichberg - Case Study Using the Composite and Proxy Design Patterns

- (c) Ordnen Sie eine Methode `delete`, die Dateien, Ordner und Verweise rekursiv löscht, einer oder mehreren geeigneten Klassen zu und implementieren Sie sie. Zeigen Sie die Löschung jedes Elements durch eine Textausgabe von `name` an. `toString()` müssen Sie dabei nicht implementieren. Gehen Sie davon aus, dass Verweis- und Ordnerstrukturen azyklisch sind und dass jedes Element des Dateisystems höchstens einmal existiert. Wenn ein Verweis gelöscht wird, wird sowohl der Verweis als auch das verwiesene Element bzw. transitiv die Kette der verwiesenen Elemente gelöscht. Bedenken Sie, dass die Löschung eines Elements immer auch Konsequenzen für den dieses Element beinhaltenden Ordner hat. Es gibt keinen Punktabzug, wenn Sie die Löschung des Root-Ordners nicht zulassen.

Lösungsvorschlag

Siehe Antwort zu Aufgabe b)

- (d) Was kann im Fall von `delete` passieren, wenn die Linkstruktur zyklisch ist oder die Ordnerstruktur zyklisch ist? Kann es zu diesen Problemen auch dann führen, wenn weder die Linkstruktur zyklisch ist, noch die Ordnerstruktur zyklisch ist? Wie kann man im Programm das Problem lösen, falls man Zyklizitäten zulassen möchte?

Falls die Link- oder Ordnerstruktur zyklisch ist, kann es aufgrund der Rekursion zu einer Endlosschleife kommen. Diese Problem tritt bei azyklischen Strukturen nicht auf, weil der rekursive Löschvorgang beim letzten Element abgebrochen wird (Abbruchbedingung).

Das Problem kann zum Beispiel durch ein neues Attribut `gelöscht` in der Klasse `Link` oder `Directory` gelöst werden. Dieses Attribut wird auf `true` gesetzt, bevor es in die Rekursion einsteigt. Rufen sich die Klassen wegen der Zyklizität selbst wieder auf, kommt es durch entsprechende IF-Bedingungen zum Abbruch.

Außerdem ist ein Zähler denkbar, der sich bei jeder Rekursion hochsetzt und ab einem gewissen Grenzwert zum Abbruch führt.

- (e) Was ist ein Design Pattern? Nennen Sie drei Beispiele und erläutern Sie sie kurz. Welches Design Pattern bietet sich für die Behandlung von hierarchischen Teil-Ganzes-Beziehungen an, wie sie im Beispiel des Dateisystems vorliegen?

Lösungsvorschlag

Design Pattern sind wiederkehrende, geprüfte, bewährte Lösungsschablonen für typische Probleme.

### Drei Beispiele

**Einzelstück (Singleton)** Stellt sicher, dass nur *genau eine Instanz einer Klasse* erzeugt wird.

**Beobachter (Observer)** Das Observer-Muster ermöglicht einem oder mehreren Objekten, automatisch auf die *Zustandsänderung* eines bestimmten Objekts zu *reagieren*, um den eigenen Zustand anzupassen.

**Stellvertreter (Proxy)** Ein Proxy stellt einen Platzhalter für eine andere Komponente (Objekt) dar und kontrolliert den Zugang zum echten Objekt.

Für hierarchischen Teil-Ganzes-Beziehungen eignet sich das Kompositum (Composite). Es ermöglicht die Gleichbehandlung von Einzelelementen und Elementgruppierungen in einer verschachtelten Struktur (z. B. Baum), sodass aus Sicht des Clients keine explizite Unterscheidung notwendig ist.

## 66116 / 2019 / Herbst / Thema 2 / Teilaufgabe 1 / Aufgabe 3

Die Komponentenarchitektur eines Softwaresystems beschreibt die unterschiedlichen Softwarebausteine, deren Schnittstellen und die Abhängigkeiten von Softwarekomponenten wie beispielsweise Klassen. Wir unterscheiden zwischen der Soll- und der Ist-Architektur eines Systems.

- (a) Nennen und definieren Sie drei Qualitätsattribute von Software, die durch die Architektur beeinflusst werden und charakterisieren Sie jeweils eine „schlechte“ Architektur, die diese Qualitätsattribute negativ beeinflusst.

**Skalierbarkeit** Definition: Ob die Software auch für große Zugriffslasten konzipiert wurde. Charakterisierung einer schlechten Architektur: Eine Anwendung läuft nur auf einem Server.

**Modifizierbarkeit** Definition: Ob die Software leicht verändert, erweitert werden kann. Charakterisierung einer schlechten Architektur: Monolithische Architektur, dass kein Laden von Modulen zulässt.

**Verfügbarkeit** Definition: Ob die Software ausfallsicher ist, im Laufenden Betrieb gewartet werden kann. Charakterisierung einer schlechten Architektur: Ein-Server-Architektur, die mehrmal neugestartet werden muss, um ein Update einzuspielen.

- (b) Erläutern Sie, was Information Hiding ist. Wie hängt Information Hiding mit Softwarearchitektur zusammen? Wie wird Information Hiding auf Klassenebene in Java implementiert? Gibt es Situationen, in denen Information Hiding auch negative Effekte haben kann?

Das Prinzip der Trennung von Zuständigkeiten (engl. separation of concerns) sorgt dafür, dass jede Komponente einer Architektur nur für eine einzige Aufgabe zuständig ist. Das Innenleben von Komponenten wird durch Schnittstellen verkapselt, was auf das Prinzip des Verbergens von Informationen (engl. information hiding) zurückgeht.

Java: Durch die Sichtbarkeits-Schlüsselwörter: private, protected

Zusätzlicher Overhead durch Schnittstellen API zwischen den Modulen.

- (c) Erklären Sie, was Refactoring ist.

Verbesserungen des Code durch bessere Lesbarkeit, Wartbarkeit, Performanz, Sicherheit. Keine neuen Funktionen werden programmiert.

- (d) Skizzieren Sie die Kernideen von Scrum inkl. der wesentlichen Prozessschritte, Artefakte und Rollen. Beschreiben Sie dann die Rolle von Ist- und Soll-Architektur in agilen Entwicklungskontexten wie Scrum.

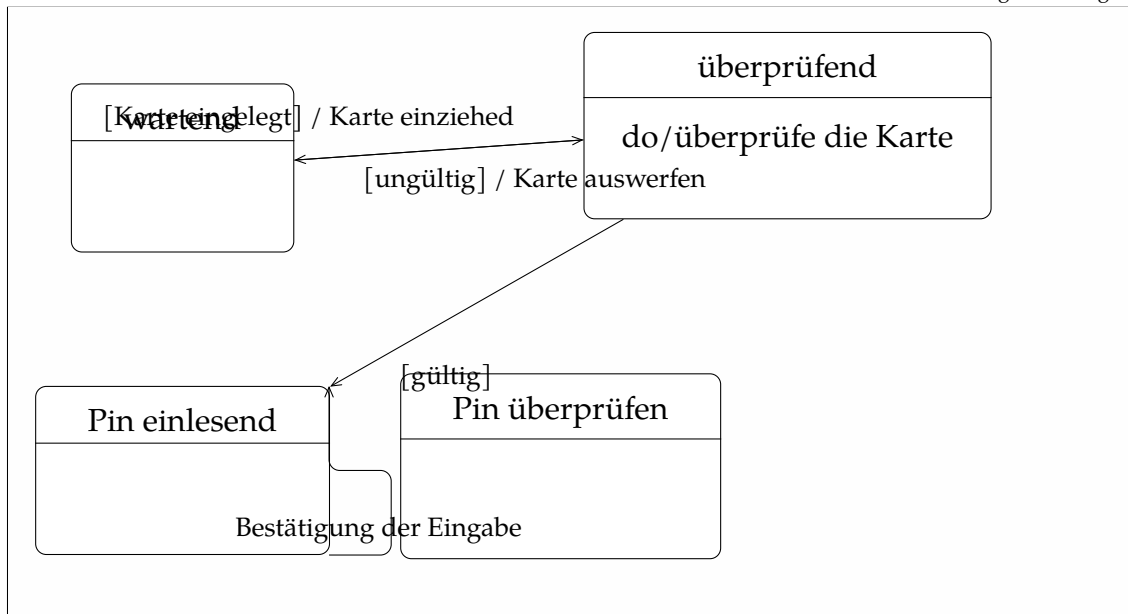
## 66116 / 2020 / Frühjahr / Thema 1 / Teilaufgabe 1 / Aufgabe 1

- (a) Basisfunktion

Erstellen Sie ein Zustandsdiagramm für einen Bankautomat, welcher den im Folgenden beschriebenen Authentifizierungsvorgang von Bankkunden realisiert. Modellieren Sie dazu soweit nötig sowohl Zustände und Transitionsbedingungen als auch die Aktionen der Zustände.

Der Bankautomat startet im Grundzustand und wartet auf das Einlegen einer Bankkarte. Wird eine Karte eingelegt, wird diese eingezogen und der Automat startet die Überprüfung der Bankkarte. Ist die Karte ungültig, wird die Karte ausgeworfen und der Automat wechselt in den Grundzustand. Ist die Karte gültig, kann die vierstellige PIN eingelesen werden. Nach der Bestätigung der Eingabe wird diese überprüft. Ist die PIN gültig, so stoppt der Automat und zeigt eine erfolgreiche Authentifizierung an. Ist die PIN ungültig, zeigt der Automat einen Fehler an und erlaubt eine erneute Eingabe der PIN.

Lösungsvorschlag



## (b) Erweiterung

Der Bankautomaten aus Aufgabe a) soll nun so verändert werden, dass ein Bankkunde nach dem ersten fehlerhaften Eingeben der PIN die PIN erneut eingeben muss. Bei erneuter Falscheingabe, wird eine dritte Eingabe möglich. Bei der dritten Falscheingabe der PIN wird die Karte vom Automaten eingezogen und der Automat geht wieder in den Ausgangszustand über.

Hinweis: Für diese Aufgabe dürfen Sie Ihr Zustandsdiagramm aus a) weiter verwenden, wenn Sie eindeutig, z. B. durch den Einsatz von Farben kennzeichnen, was nur zur Aufgabe a) gehört und was Abänderungen des Zustandsdiagramms aus a) sind. Sie können, falls Sie einen neuen Automaten zeichnen, Zustände und Übergänge, die inhaltsgleich zur Lösung des Aufgabenteils a) sind mit einem "W" markieren, statt sie zu beschriften. In diesem Fall wird der Text aus der Lösung zu Aufgabenteil a) an dieser Stelle wiederholt gedacht.

## 66116 / 2020 / Herbst / Thema 1 / Teilaufgabe 1 / Aufgabe 1

- (a) Definieren Sie die Begriffe „*partielle Korrektheit*“ und „*totale Korrektheit*“ und grenzen Sie sie voneinander ab.

**partielle Korrektheit** Ein Programmcode wird bezüglich einer Vorbedingung  $P$  und einer Nachbedingung  $Q$  partiell korrekt genannt, wenn bei einer Eingabe, die die Vorbedingung  $P$  erfüllt, jedes Ergebnis die Nachbedingung  $Q$  erfüllt. Dabei ist es noch möglich, dass das Programm nicht für jede Eingabe ein Ergebnis liefert, also nicht für jede Eingabe terminiert.

**totale Korrektheit** Ein Code wird total korrekt genannt, wenn er partiell korrekt ist und zusätzlich für jede Eingabe, die die Vorbedingung  $P$  erfüllt, terminiert. Aus der Definition folgt sofort, dass total korrekte Programme auch immer partiell korrekt sind.

- (b) Geben Sie die Verifikationsregel für die abweisende Schleife `while(b) { A }` an.

Eine abweisende Schleife ist eine while-Schleife, da die Schleifenbedingung schon bei der ersten Prüfung falsch sein kann und somit die Schleife abgewiesen wird.

Um die schwächste Vorbedingung eines Ausdrucks der Form „`while(b) { A }`“ zu finden, verwendet man eine *Schleifeninvariante*. Sie ist ein Prädikat für das

$$\{I \wedge b\}A\{I\}$$

gilt. Die Schleifeninvariante gilt also sowohl vor, während und nach der Schleife.

- (c) Erläutern Sie kurz und prägnant die Schritte zur Verifikation einer abweisenden Schleife mit Vorbedingung  $P$  und Nachbedingung  $Q$ .

**Schritt 0:** Schleifeninvariante  $I$  finden

**Schritt 1:**  $I$  gilt vor Schleifenbeginn,

d.h.  $P \Rightarrow \text{wp}(\text{"Code vor Schleife"}, I)$

**Schritt 2:**  $I$  gilt nach jedem Schleifendurchlauf

d.h.  $I \wedge b \Rightarrow \text{wp}(\text{"Code in der Schleife"}, I)$

**Schritt 3:** Bei Terminierung der Schleife liefert Methode das gewünschte Ergebnis,

d.h.  $I \wedge \neg b \Rightarrow \text{wp}(\text{"Code nach der Schleife"}, Q)$

- (d) Wie kann man die Terminierung einer Schleife beweisen?

Zum Beweis der Terminierung einer Schleife muss eine Terminierungsfunktion  $T$  angegeben werden:

$$T: V \rightarrow \mathbb{N}$$

$V$  ist eine Teilmenge der Ausdrücke über die Variablenwerte der Schleife

Die Terminierungsfunktion muss folgende Eigenschaften besitzen:

- Ihre Werte sind natürliche Zahlen (einschließlich 0).
- Jede Ausführung des Schleifenrumpfs verringert ihren Wert (streng monoton fallend).
- Die Schleifenbedingung ist false, wenn  $T = 0$ .

$T$  ist die obere Schranke für die noch ausstehende Anzahl von Schleifendurchläufen.

Beweise für Terminierung sind nicht immer möglich!



- (e) Geben Sie für das folgende Suchprogramm die nummerierten Zusicherungen an. Lassen Sie dabei jeweils die invariante Vorbedingung  $P$  des Suchprogramms weg. Schreiben Sie nicht auf dem Aufgabenblatt!

$$P \equiv n > 0 \wedge a_0 \dots a_{n-1} \in \mathbb{Z}^n \wedge m \in \mathbb{Z}$$

```

int i = -1;
// (1)
int j = 0;
// (2)
while (i == -1 && j < n) // (3)
{ // (4)
    if (a[j] == m) {
        // (5)
        i = j;
        // (6)
    } else {
        // (7)
        j = j + 1;
        // (8)
    }
    // (9)
}

```

Code-Beispiel auf Github ansehen: [src/main/java/org/beschlangaul/examen/examen\\_66116/jahr\\_2020/herbst/Verifikation.java](https://github.com/src/main/java/org/beschlangaul/examen/examen_66116/jahr_2020/herbst/Verifikation.java)

$$Q \equiv P \wedge (i = -1 \wedge \forall 0 \leq k < n: a_k \neq m) \vee (i \geq 0 \wedge a_i = m)$$

Lösungsvorschlag

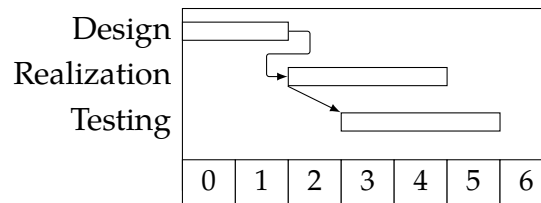
In dieser Aufgabenstellung fällt die Vorbedingung mit der Invariante zusammen. Es muss kein wp-Kalkül berechnet werden, sondern „nur“ die Zuweisungen nachverfolgt werden, um zum Schluss die Nachbedingung zu erhalten.

1.  $(i = -1) \wedge P$
2.  $(i = -1) \wedge (j = 0) \wedge P$
3.  $(i = -1) \wedge (0 \leq j < n) \wedge P$
4.  $(i = -1) \wedge (0 \leq j < n) \wedge P$
5.  $(i = -1) \wedge (a_j = m) \wedge (0 \leq j < n) \wedge P$
6.  $(i \geq 0) \wedge i = j \wedge (a_j = m) \wedge (0 \leq j < n) \wedge P$
7.  $(i = -1) \wedge (\forall 0 \leq j < n: a_j \neq m) \wedge P$
8.  $(i = -1) \wedge (\forall 0 < j \leq n: a_j \neq m) \wedge P$
9.  $((i = -1) \wedge (\forall 0 \leq k < j: a_k \neq m)) \vee ((i \geq 0) \wedge (a_i = m)) \wedge P$

## 66116 / 2020 / Herbst / Thema 1 / Teilaufgabe 1 / Aufgabe 2

Die Planung eines Softwareprojekts kann z. B. in Form von Gantt-Diagrammen oder CPM-Netzwerken (kritischer Pfad Methode) festgehalten werden.

Folgendes Gantt-Diagramm zeigt einen Teil der Projektplanung in einem klassischen Softwareentwicklungsprozess:



- (a) Im Diagramm werden 3 Phasen aus dem klassischen Softwareentwicklungsprozess genannt. Welche Phase sollte dem Design (Entwurf) immer vorangehen?

Lösungsvorschlag

Die Anforderungsdefinition

- (b) Wandeln Sie das Gantt-Diagramm in ein CPM-Netzwerk um. Fügen Sie dazu einen zusätzlichen Start- und Endknoten hinzu. Das Ende des Projekts ist durch das Ende aller Aktivitäten bedingt.

Lösungsvorschlag

$D_A$  Design Anfang

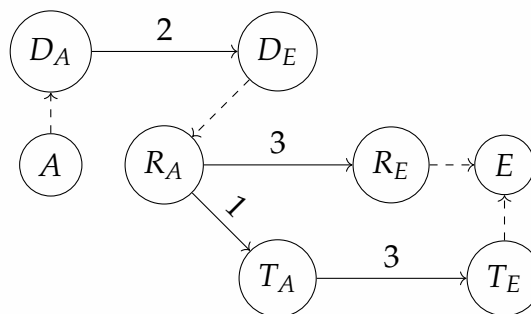
$R_A$  Realization Anfang

$T_A$  Testing Anfang

$D_E$  Design Ende

$R_E$  Realization Ende

$T_E$  Testing Ende



- (c) Welche im obigen Gantt-Diagramm nicht enthaltenen Beziehungsarten zwischen Aktivitäten können in einem Gantt-Diagramm noch auftreten? Nennen Sie auch deren Bedeutung.

Lösungsvorschlag

Diese Beziehungsarten sind im obigen Gantt-Diagramm vorhanden:

**Normalfolge EA:** *end-to-start relationship* Anordnungsbeziehung vom Ende eines Vorgangs zum Anfang seines Nachfolgers.

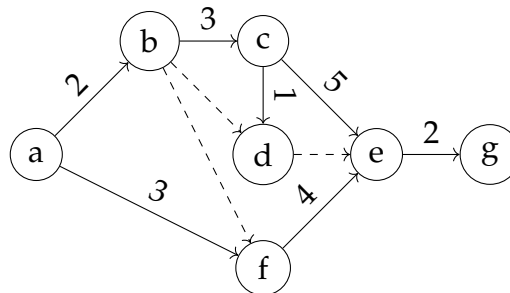
**Anfangsfolge AA:** *start-to-start relationship* Anordnungsbeziehung vom Anfang eines Vorgangs zum Anfang seines Nachfolgers.

Diese Beziehungsarten sind im obigen Gantt-Diagramm *nicht* vorhanden:

**Endefolge EE:** *finish-to-finish relationship* Anordnungsbeziehung vom Ende eines Vorgangs zum Ende seines Nachfolgers.

**Sprungfolge AE:** *start-to-finish relationship* Anordnungsbeziehung vom Anfang eines Vorgangs zum Ende seines Nachfolgers

Gegeben sei nun das folgende CPM-Netzwerk:



(d) Geben Sie für jedes Ereignis die früheste Zeit an.

Lösungsvorschlag

— Wir führen eine Vorwärtsterminierung durch und addieren die Dauern. Kann ein Ereignis über mehrere Vorgänge erreicht werden, wählen wir das Maximum aus. **Erläuterungen:**  $i$ : Ereignis  $i$ ;  $FZ_i$ : Frühester Zeitpunkt, zu dem Ereignis  $i$  eintreten kann.

$i$	Nebenrechnung	$FZ_i$
a		0
b		2
c		5
d	$\max(2_b, 6_c)$	6
e	$\max(6_d, 10_e, 7_f)$	10
f	$\max(3_f, 2_b)$	3
g		12

(e) Geben Sie für jedes Ereignis die späteste Zeit an.

Lösungsvorschlag

— Wir führen eine Rückwärtsterminierung durch und subtrahieren die Dauern vom letzten Ereignis aus. Kann ein Ereignis über mehrere Vorgänge erreicht werden, wählen wir das Minimum aus. **Erläuterungen:**  $i$ : Ereignis  $i$ ;

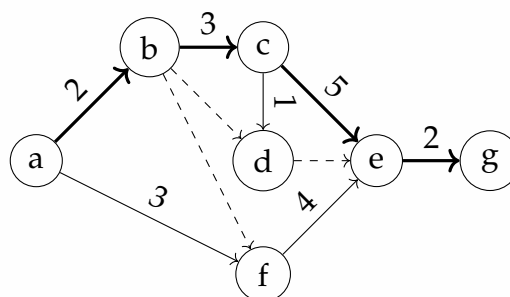
$SZ_i$ : Spätester Zeitpunkt, zu dem Ereignis  $i$  eintreten kann. \_\_\_\_\_

$i$	Nebenrechnung	$SZ_i$
g		12
f		6
e		10
d		10
c	$\min(9_d, 5_e)$	5
b	$\min(2_c, 10_d, 6_f)$	2
a		0

- (f) Geben Sie einen kritischen Pfad durch das Netz an! Wie wirkt sich eine Verzögerung von 5 Zeiteinheiten auf dem kritischen Pfad auf das Projektende aus?

Lösungsvorschlag

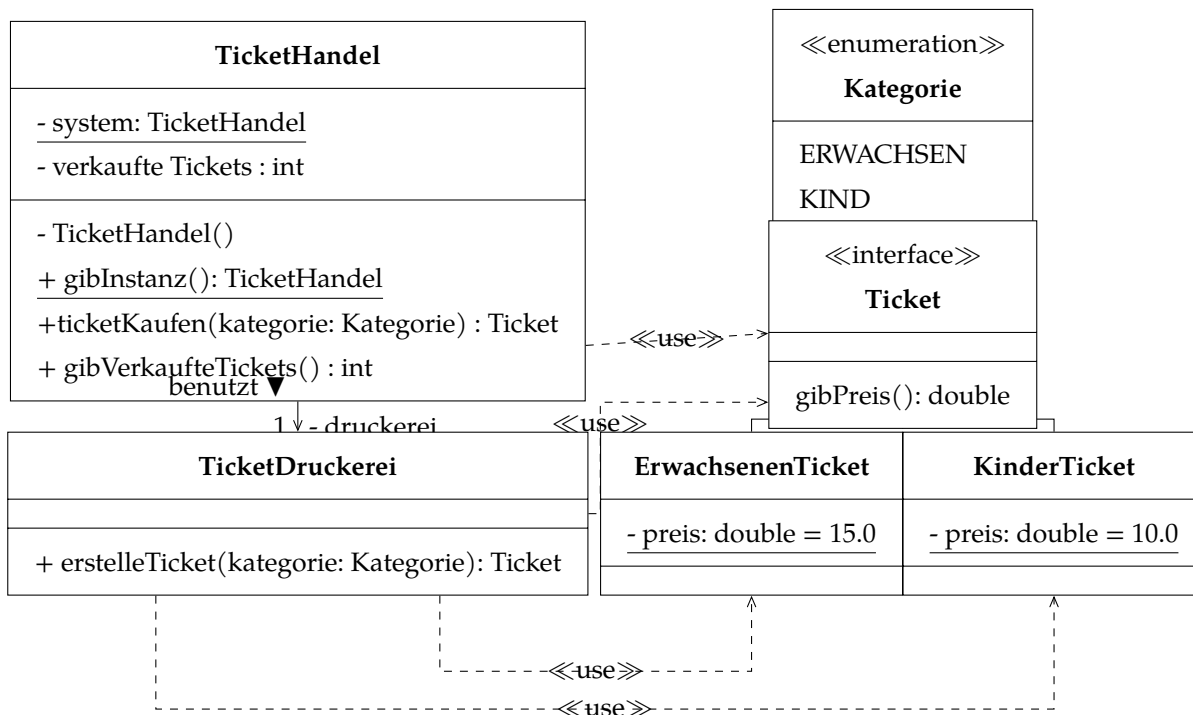
$i$	a	b	c	d	e	f	g
$FZ_i$	0	2	5	6	10	3	12
$SZ_i$	0	2	5	10	10	6	12
GP	0	0	0	3	0	3	0



Kritischer Pfad:  $a \rightarrow b \rightarrow c \rightarrow e \rightarrow g$

Das Projekt verlängert sich um 5 Zeiteinheiten.

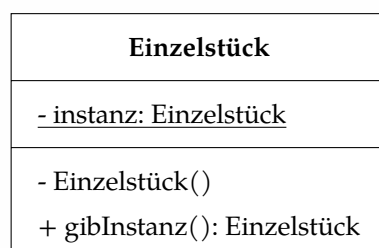
## 66116 / 2020 / Herbst / Thema 1 / Teilaufgabe 1 / Aufgabe 3



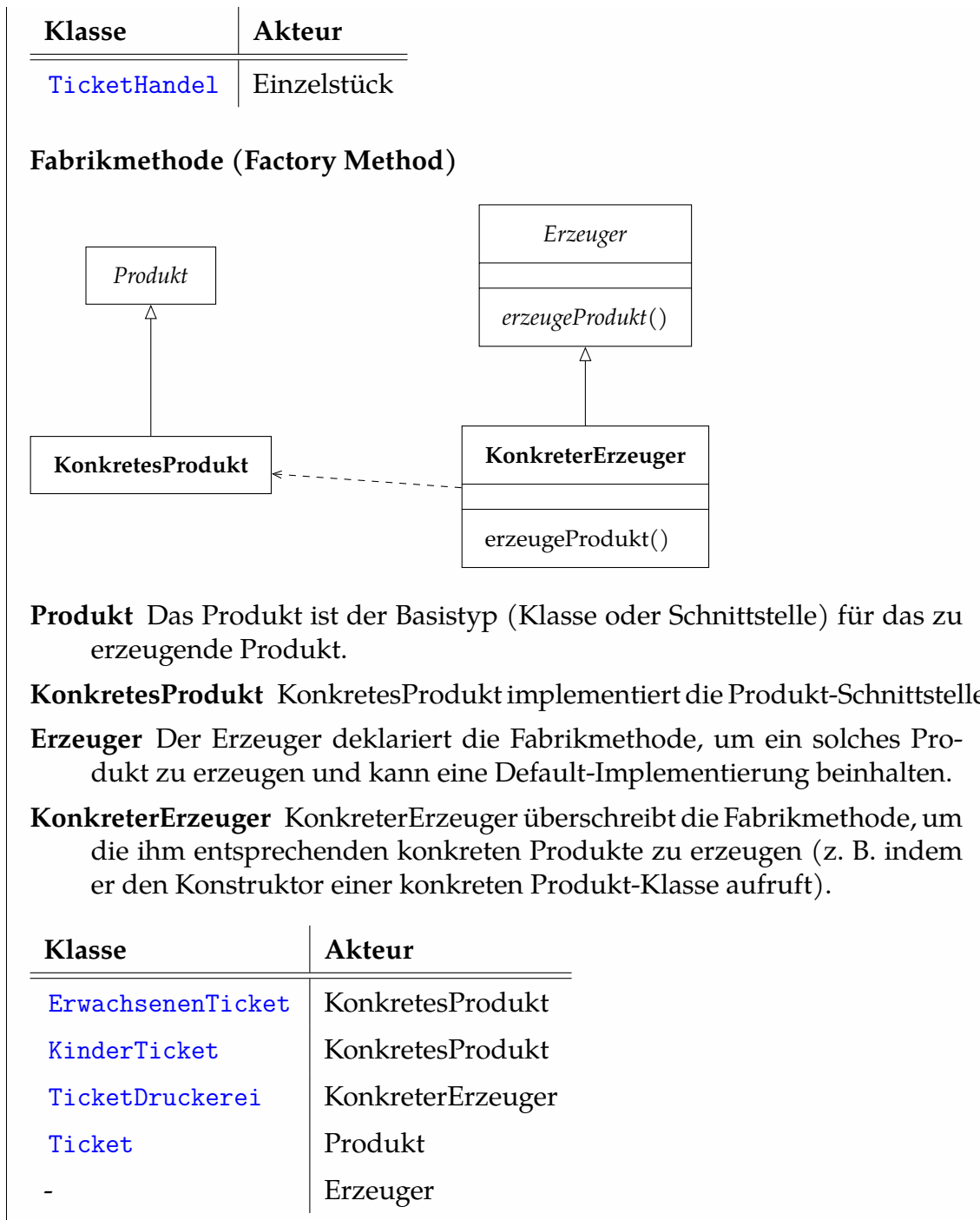
Ihnen sei ein UML-Klassendiagramm zu folgendem Szenario gegeben. Ein Benutzer (nicht im Diagramm enthalten) kann über einen **TicketHandel** Tickets erwerben. Dabei muss der Benutzer eine der zwei Ticketkategorien angeben. Das Handelsystem benutzt eine **TicketDruckerei**, um ein passendes **Ticket** für den Benutzer zu erzeugen.

- (a) Im angegebenen Klassendiagramm wurden zwei unterschiedliche Entwurfsmuster verwendet. Um welche Muster handelt es sich? Geben Sie jeweils den Namen des Musters sowie die Elemente des Klassendiagramms an, mit denen diese Muster im Zusammenhang stehen. ACHTUNG: Es handelt sich dabei *nicht* um das *Interface-* oder das *Vererbungsmuster*.

Lösungsvorschlag

**Einzelstück (Singleton)**

**Einzelstück (Singleton)** stellt eine statische Methode bereit, mit deren Hilfe die Klienten nur auf eine einzige Instanz der Klasse zugreifen können.



(b) Nennen Sie zwei generelle Vorteile von Entwurfsmustern.

Lösungsvorschlag

- Wiederverwendung einer bewährten Lösung für eine bestimmte Problemstellungen
- Verbesserung der Kommunikation unter EntwicklerInnen

(c) Geben Sie eine Implementierung der Klasse [TicketHandel](#) an. Bei der Methode [ticketKaufen\(\)](#) wird die Anzahl der verkauften Tickets um 1 erhöht und ein

entsprechendes Ticket erstellt und zurückgegeben. Beachten Sie den Hinweis auf der nächsten Seite.

Lösungsvorschlag

```
public class TicketHandel {
    private static TicketHandel system;
    private int verkaufteTickets;
    private TicketDruckerei druckerei;

    private TicketHandel() {
        druckerei = new TicketDruckerei();
        verkaufteTickets = 0;
    }

    public static TicketHandel gibInstanz() {
        if (system == null) {
            system = new TicketHandel();
        }
        return system;
    }

    public Ticket ticketKaufen(Kategorie kategorie) {
        verkaufteTickets++;
        return druckerei.erstelleTicket(kategorie);
    }

    public int gibVerkaufteTickets() {
        return verkaufteTickets;
    }
}
```

Code-Beispiel auf Github ansehen: [src/main/java/org/bschlangaul/examen/examen\\_66116/jahr\\_2020/herbst/ticket/TicketHandel.java](https://github.com/bschlangaul/examen/examen_66116/jahr_2020/herbst/ticket/TicketHandel.java)

(d) Geben Sie eine Implementierung der Klasse `TicketDruckerei` an.

Lösungsvorschlag

```
public class TicketDruckerei {
    public Ticket erstelleTicket(Kategorie kategorie) {
        if (kategorie == Kategorie.ERWACHSENEN) {
            return new ErwachsenenTicket();
        } else {
            return new KinderTicket();
        }
    }
}
```

Code-Beispiel auf Github ansehen: [src/main/java/org/bschlangaul/examen/examen\\_66116/jahr\\_2020/herbst/ticket/TicketDruckerei.java](https://github.com/bschlangaul/examen/examen_66116/jahr_2020/herbst/ticket/TicketDruckerei.java)

(e) Geben Sie eine Implementierung der Klasse `KinderTicket` an.

Lösungsvorschlag

```
public class KinderTicket implements Ticket {
    private static double preis = 10.0;
```

```
public double gibPreis() {  
    return preis;  
}
```

Code-Beispiel auf Github ansehen: [src/main/java/org/bschlangaul/examen/examen\\_66116/jahr\\_2020/herbst/ticket/KinderTicket.java](https://github.com/bschlangaul/examen/examen_66116/jahr_2020/herbst/ticket/KinderTicket.java)

*Hinweis:* Die Implementierungen *müssen* sowohl dem Klassendiagramm, als auch den Konzepten der verwendeten Muster entsprechen. Verwenden Sie eine objekt-orientierte Programmiersprache, vorzugsweise *Java*. Sie müssen sich an der nachfolgenden Testmethode und ihrer Ausgabe orientieren. Die Testmethode muss mit Ihrer Implementierung ausführbar sein und sich semantisch korrekt verhalten.

Quelltext der Testmethode:

```
public static void main(String[] args) {  
    TicketHandel.gibInstanz().ticketKaufen(Kategorie.ERWACHSEN);  
    TicketHandel.gibInstanz().ticketKaufen(Kategorie.KIND);  
    System.out.println("Anzahl verkaufter Tickets: " +  
    TicketHandel.gibInstanz().gibVerkaufteTickets());  
}
```

Code-Beispiel auf Github ansehen: [src/main/java/org/bschlangaul/examen/examen\\_66116/jahr\\_2020/herbst/ticket/Test.java](https://github.com/bschlangaul/examen/examen_66116/jahr_2020/herbst/ticket/Test.java)

Konsolenausgabe:

Anzahl verkaufter Tickets: 2

## 66116 / 2020 / Herbst / Thema 1 / Teilaufgabe 1 / Aufgabe 4

Diese Aufgabe behandelt *Wortpalindrome*, also Wörter, die vorwärts und rückwärts gelesen jeweils dieselbe Zeichenkette bilden, z. B. Otto oder Rentner. Leere Wortpalindrome (also Wortpalindrome der Wortlänge 0) sind dabei nicht zulässig.

Folgende *Java-Methode* prüft, ob das übergebene Zeichen-Array ein Wortpalindrom darstellt:

```
public static boolean istWortpalindrom(char[] wort) { // 1  
    boolean resultat = false; // 2  
    if (wort != null) { // 3  
        int laenge = wort.length; // 4  
        if (laenge >= 2) { // 5  
            resultat = true; // 6  
            for (int i = 0; i < laenge / 2; ++i) { // 7  
                char c1 = wort[i]; // 8  
                char c2 = wort[laenge - 1 - i]; // 9  
                if (Character.toLowerCase(c1) != Character.toLowerCase(c2)) // 10  
                { // 11  
                    resultat = false; // 12  
                    break; // 13  
                } // 14  
            } // 15  
        }  
    }  
}
```

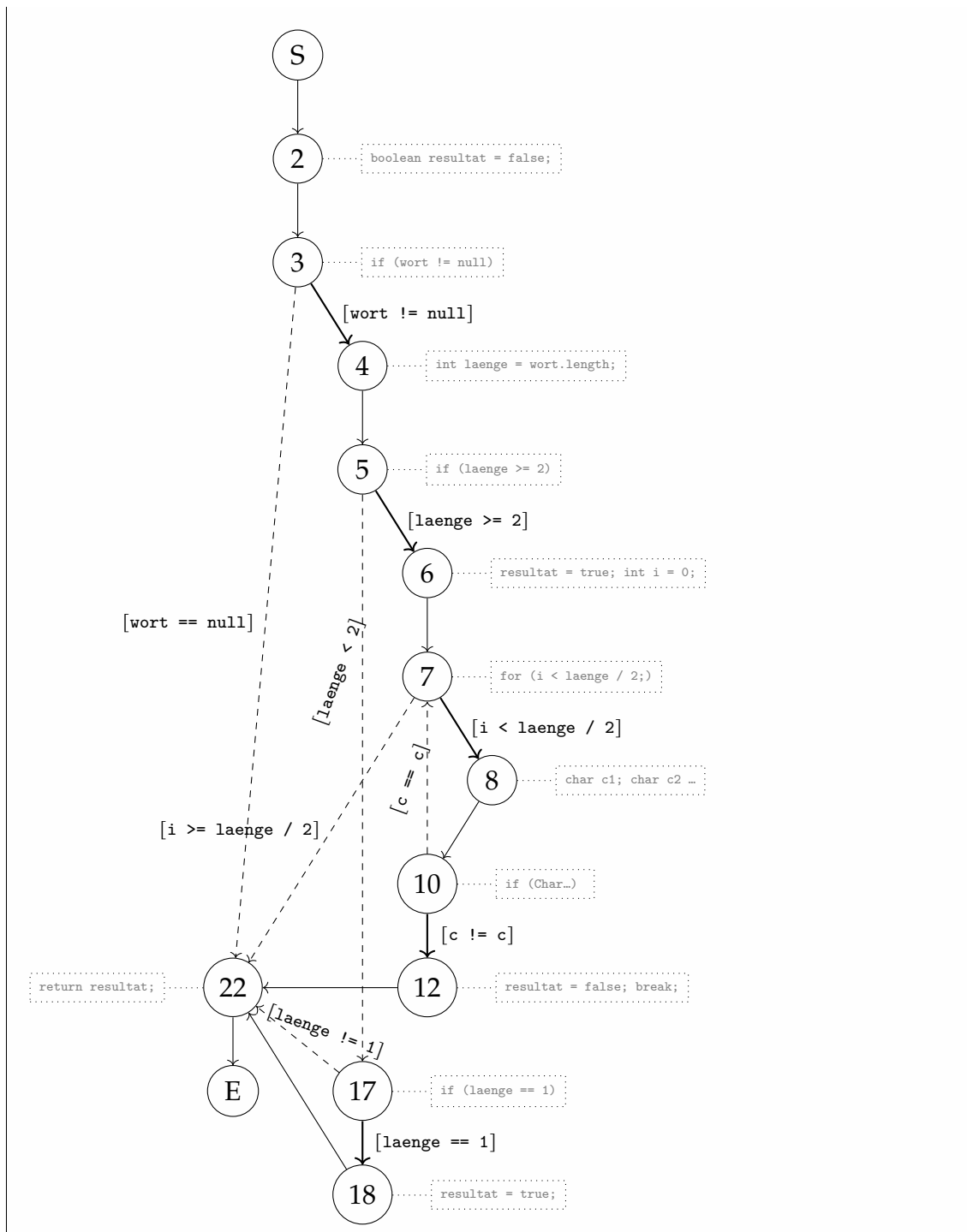


```
    } else { // 16
      if (laenge == 1) { // 17
        resultat = true; // 18
      } // 19
    } // 20
  } // 21
  return resultat; // 22
} // 23
```

Code-Beispiel auf Github ansehen: [src/main/java/org/bschlangaul/examen/examen\\_66116/jahr\\_2020/herbst/Palindrom.java](https://github.com/bschlangaul/examen/examen_66116/jahr_2020/herbst/Palindrom.java)

- (a) Geben Sie für die Methode einen *Kontrollflussgraphen* an, wobei Sie die Knoten mit den jeweiligen Zeilennummern im Quelltext beschriften.

Lösungsvorschlag



- (b) Geben Sie eine *minimale Testmenge* an, die das Kriterium der Anweisungsüberdeckung erfüllt.

Hinweis: Eine *Testmenge* ist *minimal*, wenn es keine Testmenge mit einer kleineren Zahl von Testfällen gibt. Die Minimalität muss *nicht* bewiesen werden.

```
isWortpalindrom(new char[] { 'a' }):
```

Ⓢ - ② - ③ - ④ - ⑤ - ⑰ - ⑱ - ㉒ - Ⓔ

```
isWortpalindrom(new char[] { 'a', 'b' }):
```

(S) - (2) - (3) - (4) - (5) - (6) - (7) - (8) - (10) - (12) - (22) - (E)

- (c) Geben Sie eine *minimale Testmenge* an, die das Kriterium der *Boundary-Interior-Pfadüberdeckung* erfüllt.

Hinweis: Das Kriterium *Boundary-Interior-Pfadüberdeckung* beschreibt einen Spezialfall der Pfadüberdeckung, wobei nur Pfade berücksichtigt werden, bei denen jede Schleife nicht mehr als zweimal durchlaufen wird.

Lösungsvorschlag

Es gibt noch ganz viele infeasible Pfade, die hier nicht aufgeführt werden.

### Äußere Pfade

```
- isWortpalindrom(null):
```

(S) - (2) - (3) - (22) - (E)

```
- isWortpalindrom(new char[] { }):
```

(S) - (2) - (3) - (4) - (5) - (17) - (22) - (E)

```
- isWortpalindrom(new char[] { 'a' }):
```

(S) - (2) - (3) - (4) - (5) - (17) - (18) - (22) - (E)

**Grenzpfade (boundary paths, boundary test)** Für Schleifen fester Lauflänge ist diese Testfallgruppe leer.

```
isWortpalindrom(new char[] { 'a', 'a' }):
```

(S) - (2) - (3) - (4) - (5) - (6) - (7) - (8) - (10) - (7) - (22) - (E)

```
isWortpalindrom(new char[] { 'a', 'b' }):
```

(S) - (2) - (3) - (4) - (5) - (6) - (7) - (8) - (10) - (12) - (22) - (E)

### Innere Pfade (interior test)

```
- isWortpalindrom(new char[] { 'a', 'a', 'a', 'a' }):
```

(S) - (2) - (3) - (4) - (5) - (6) - (7) - (8) - (10) - (7) - (8) - (10) - (7) - (22) - (E)

```
- isWortpalindrom(new char[] { 'a', 'b', 'a', 'a' }):
```

(S) - (2) - (3) - (4) - (5) - (6) - (7) - (8) - (10) - (7) - (8) - (10) - (12) - (22) - (E)

- (d) Im Falle des Kriteriums Pfadüberdeckung können minimale Testmengen sehr groß werden, da die Anzahl der Pfade sehr schnell zunimmt. Wie viele *mögliche Pfade* ergeben sich maximal für eine Schleife, die drei einseitig bedingte Anweisungen hintereinander enthält und bis zu zweimal durchlaufen wird? Geben Sie Ihren Rechenweg an (das Ergebnis alleine gibt keine Punkte).

Lösungsvorschlag

Pro Schleifendurchlauf:  $2 \cdot 2 \cdot 2 = 2^3 = 8$

Maximal 2 Schleifendurchläufe:  $2 \cdot 8 = 16$

- (e) Könnte für das hier abgebildete Quelltext-Beispiel auch das Verfahren der *unbegrenzten Pfadüberdeckung* (also Abdeckung aller möglicher Pfade ohne Beschränkung) als Test-Kriterium gewählt werden? Begründen Sie.

Lösungsvorschlag

Kante 7 nach 22

**66116 / 2020 / Herbst / Thema 1 / Teilaufgabe 1 / Aufgabe 5**

- (a) Nennen Sie fünf kritische Faktoren, die bei der Auswahl eines Vorgehensmodells helfen können und ordnen Sie plangetriebene und agile Prozesse entsprechend ein.

Lösungsvorschlag

- (i) Vollständigkeit der Anforderungen
  - bei vollständiger Kenntnis der Anforderungen: *plangetrieben*
  - bei teilweiser Kenntnis der Anforderungen: *agil*
- (ii) Möglichkeit der Rücksprache mit dem Kunden
  - keine Möglichkeit: *plangetrieben*
  - Kunde ist partiell immer wieder involviert: *agil*
- (iii) Teamgröße
  - kleine Teams (max. 10 Personen): *agil*
  - größere Teams: *plangetrieben*
- (iv) Bisherige Arbeitsweise des Teams
  - bisher feste Vorgehensmodelle: *plangetrieben*
  - flexible Arbeitsweisen: *agil*
- (v) Verfügbare Zeit
  - kurze Zeitvorgabe: *plangetrieben*
  - möglichst schnell funktionierender Prototyp verlangt: *agil*
  - beide Vorgehensmodelle sind allerdings zeitlich festgelegt

Mögliche weitere Faktoren: Projektkomplexität, Dokumentation

- (b) Nennen und beschreiben Sie kurz die Rollen im Scrum.

Lösungsvorschlag

**Product Owner** Der Product Owner ist für die Eigenschaften und den wirtschaftlichen Erfolg des Produkts verantwortlich.

**Entwickler** Die Entwickler sind für die Lieferung der Produktfunktionalitäten in der vom Product Owner gewünschten Reihenfolge verantwortlich.

**Scrum Master** Der Scrum Master ist dafür verantwortlich, dass Scrum als Rahmenwerk gelingt. Dazu arbeitet er mit dem Entwicklungsteam zusammen, gehört aber selbst nicht dazu.

- (c) Nennen und beschreiben Sie drei Scrum Artefakte. Nennen Sie die verantwortliche Rolle für jedes Artefakt.

Lösungsvorschlag

**Product Backlog** Das Product Backlog ist eine geordnete Auflistung der Anforderungen an das Produkt.

**Sprint Backlog** Das Sprint Backlog ist der aktuelle Plan der für einen Sprint zu erledigenden Aufgaben.

**Product Increment** Das Inkrement ist die Summe aller Product-Backlog-Einträge, die während des aktuellen und allen vorangegangenen Sprints fertiggestellt wurden.

- (d) Beschreiben Sie kurz, was ein Sprint ist. Wie lange sollte ein Sprint maximal dauern?

Lösungsvorschlag

Ein Sprint ist ein Arbeitsabschnitt, in dem ein Inkrement einer Produktfunktionalität implementiert wird. Ein Sprint umfasst ein Zeitfenster von ein bis vier Wochen.

## 66116 / 2020 / Herbst / Thema 2 / Teilaufgabe 1 / Aufgabe 1

- (a) Nennen Sie fünf Phasen, die im Wasserfallmodell durchlaufen werden sowie deren jeweiliges Ziel bzw. Ergebnis(-dokument).

Lösungsvorschlag

**Anforderung** Lasten- und Pflichtenheft

**Entwurf** Softwarearchitektur in Form von Struktogrammen, UML-Diagrammen etc.

**Implementierung** Software

**Überprüfung** überarbeitete Software

**Einsatz und Wartung** erneut überarbeitete Software, verbesserte Wartung

- (b) Nennen Sie drei Arten der Softwarewartung und geben Sie jeweils eine kurze Beschreibung an.

Lösungsvorschlag

**korrektive Wartung** Korrektur von Fehlern, die schon beim Kunden in Erscheinung getreten sind

**präventive Wartung** Korrektur von Fehlern, die beim Kunden noch nicht in Erscheinung getreten sind

**adaptive Wartung** Anpassung der Software an neue Anforderungen

**perfektionierende Wartung** Verbesserung von Performance und Wartbarkeit

und Behebung von technical debts<sup>a</sup>

<sup>a</sup><https://de.wikipedia.org/wiki/Softwarewartung>

Kontinuierliche Integration  
(Continuous Integration)  
EXtreme Programming  
Softwaremaße  
SCRUM

- (c) Eine grundlegende Komponente des *Extreme Programming* ist „*Continuous Integration*“. Erklären Sie diesen Begriff und warum man davon einen Vorteil erwartet.

Lösungsvorschlag

Die Software wird nach jeder Änderung (push) automatisch kompiliert und auch getestet. Man erwartet sich davon einen Vorteil, weil Fehler schneller erkannt werden und die aktuelle Version des Systems ständig verfügbar ist.

- (d) Nennen Sie zwei Softwaremetriken und geben Sie jeweils einen Vor- und Nachteil an, der im Projektmanagement von Bedeutung ist.

Lösungsvorschlag

### Anzahl der Code-Zeilen

**Vorteil:** Ist sehr einfach zu bestimmen.

**Nachteil:** Entwickler können die Zeilenzahl manipulieren (zusätzliche Leerzeilen, Zeilen aufteilen), ohne dass sich die Qualität des Codes verändert.

### Zyklomatische Komplexität

**Vorteil:** Trifft eine Aussage über die Qualität des Algorithmus.

**Nachteil:** Ist für Menschen nicht intuitiv zu erfassen. Zwei Codes, die dasselbe Problem lösen können die gleiche Zyklomatische Komplexität haben, obwohl der eine wesentlich schlechter zu verstehen ist (Spaghetticode!).

- (e) Nennen und beschreiben Sie kurz drei wichtige Aktivitäten, welche innerhalb einer Sprint-Iteration von Scrum durchlaufen werden.

Lösungsvorschlag

**Sprint Planning** Im Sprint Planning werden zwei Fragen beantwortet:

- Was kann im kommenden Sprint entwickelt werden?
- Wie wird die Arbeit im kommenden Sprint erledigt?

Die Sprint-Planung wird daher häufig in zwei Teile geteilt. Sie dauert in Summe maximal 2 Stunden je Sprint-Woche, beispielsweise maximal acht Stunden für einen 4-Wochen-Sprint.

**Daily Scrum** Zu Beginn eines jeden Arbeitstages trifft sich das Entwicklerteam zu einem max. 15-minütigen Daily Scrum. Zweck des Daily Scrum ist der Informationsaustausch.

**Sprint Review** Das Sprint Review steht am Ende des Sprints. Hier überprüft das Scrum-Team das Inkrement, um das Product Backlog bei Bedarf anzupassen. Das Entwicklungsteam präsentiert seine Ergebnisse und es wird überprüft, ob das zu Beginn gesteckte Ziel erreicht wurde.

- (f) Erläutern Sie den Unterschied zwischen dem Product-Backlog und dem Sprint-Backlog.

Lösungsvorschlag

**Product Backlog** Das Product Backlog ist eine geordnete Auflistung der Anforderungen an das Produkt.

**Sprint Backlog** Das Sprint Backlog ist der aktuelle Plan der für einen Sprint zu erledigenden Aufgaben.

- (g) Erläutern Sie, warum eine agile Entwicklungsmethode nur für kleinere Teams (max. 10 Personen) gut geeignet ist, und zeigen Sie einen Lösungsansatz, wie auch größere Firmen agile Methoden sinnvoll einsetzen können.

Lösungsvorschlag

Bei agilen Entwicklungsmethoden finden daily scrums statt, die in der intendierten Kürze nur in kleinen Teams umsetzbar sind. Außerdem ist in größeren Teams oft eine Hierarchie vorhanden, die eine schnelle Entscheidungsfindung verhindert. Man könnte die große Firma in viele kleine Teams einteilen, die jeweils an kleinen (Teil-)Projekten arbeiten. Eventuell können die product owner der einzelnen Teams nach agilen Methoden zusammenarbeiten.

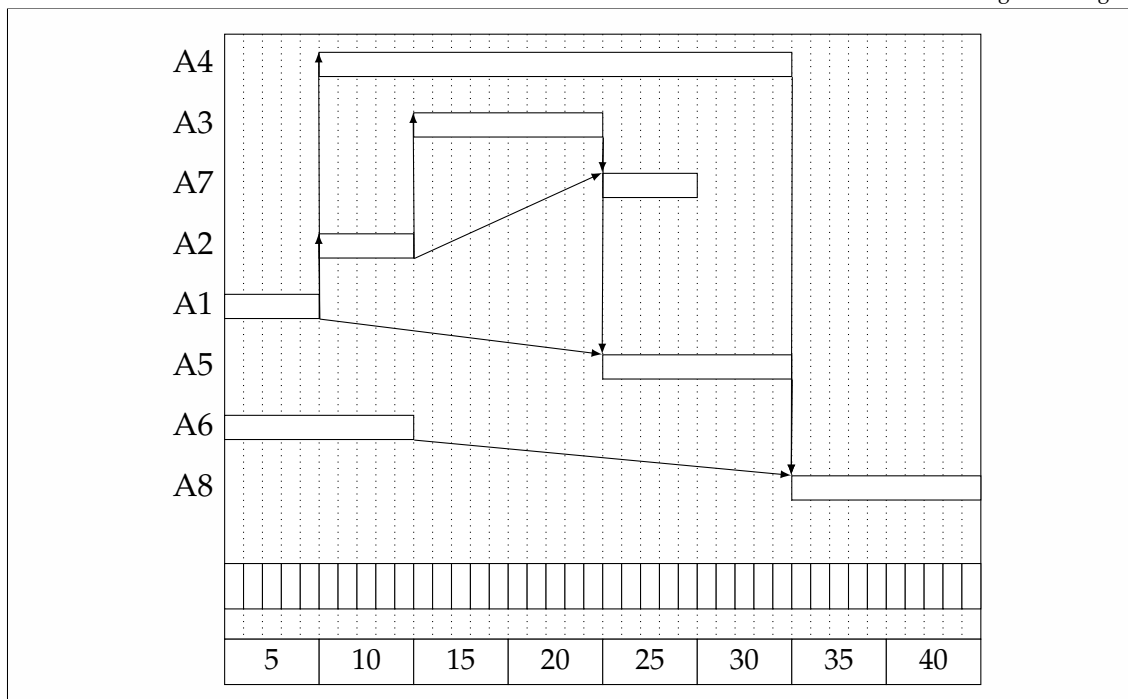
## 66116 / 2020 / Herbst / Thema 2 / Teilaufgabe 1 / Aufgabe 2

Betrachten Sie die folgende Tabelle zum Projektmanagement:

Arbeitspaket	Dauer (Tage)	abhängig von
A1	5	
A2	5	A1
A3	10	A2
A4	25	A1
A5	10	A1, A3
A6	10	
A7	5	A2, A3
A8	10	A4, A5, A6

- (a) Erstellen Sie ein Gantt-Diagramm, das die in der Tabelle angegebenen Abhängigkeiten berücksichtigt. Das Diagramm muss nicht maßstabsgetreu sein, jedoch jede Information aus der gegebenen Tabelle enthalten.





(b) Wie lange dauert das Projekt mindestens?

Lösungsvorschlag

Es dauert mindestens 40 Tage.

(c) Geben Sie alle kritischen Pfade an.

Lösungsvorschlag

A1 → A4 → A8

A1 → A2 → A3 → A5 → A8

(d) Bewerten Sie folgende Aussage eines Projektmanagers: „Falls unser Projekt in Verzug gerät, bringen uns neue Programmierer auch nicht weiter.“

Lösungsvorschlag

Ist der Verzug in einem Arbeitspaket, in dem genügend Pufferzeit vorhanden ist (hier A6), so hilft ein neuer Programmierer nicht unbedingt weiter. Geht allerdings die Pufferzeit zu Ende oder ist erst gar nicht vorhanden (z. B. im kritischen Pfad), so kann ein/e neue/r ProgrammierIn helfen, falls deren/dessen Einarbeitungszeit gering ist. Muss sich der/die Neue erste komplett einarbeiten, so wird er/sie wohl auch keine große Hilfe sein.

## 66116 / 2020 / Herbst / Thema 2 / Teilaufgabe 1 / Aufgabe 4

Betrachten Sie das folgende Szenario:

Entwickeln Sie für einen Kunden eine einheitliche Online-Plattform, in welcher mehrere Restaurants angebunden sind. Die Nutzer des Systems sollen Essen wie Pizzen, Burger oder Pasta bestellen und dabei aus einer Liste von verschiedenen Gerichten auswählen können. Die Plattform soll zusätzliche Optionen (z. B. Lieferung durch einen Lieferdienst oder direkte Abholung, inkl. Salat oder einer Flasche Wein) ermöglichen. Die Bestellung soll dann von der Plattform an den jeweiligen Gaststättenbetreiber gesendet werden. Die Besteller sollen zudem eine Bestätigung als Nachricht erhalten. Die jeweiligen Gerichte und Optionen haben unterschiedliche Preise, die dem Internetnutzer angezeigt werden müssen, bevor er diese auswählt. Der Endpreis muss vor der endgültigen Zahlung des Auftrags angezeigt werden. Kunden können (optional) einen Benutzeraccount anlegen und erhalten bei häufigen Bestellungen einen Rabatt.

- (a) Beschreiben Sie kurz ein Verfahren, wie Sie aus der Szenariobeschreibung mögliche Kandidaten für Klassen erhalten können.

Lösungsvorschlag

Verfahren nach Abbott;<sup>a</sup>

Objektorientierte Analyse und Design (OOAD)

<sup>a</sup>[http://info.johpie.de/stufe\\_q1/info\\_01\\_verfahren\\_abbott.pdf](http://info.johpie.de/stufe_q1/info_01_verfahren_abbott.pdf)

- (b) Beschreiben Sie kurz ein Verfahren, um Vererbungshierarchien zu identifizieren.

Lösungsvorschlag

Eine Begriffshierarchie mit den Substantiven des Textes bilden.

- (c) Geben Sie fünf geeignete Klassen für das obige Szenario an. Nennen Sie dabei keine Klassen, welche durch Basisdatentypen wie Integer oder String abgedeckt werden können.

Lösungsvorschlag

- Benutzer (Kunde, Gaststättenbetreiber)
- Restaurant
- Gericht (Pizza, Burger, Pasta)
- ZusatzOption (Lieferung, Salat, Wein)
- Bestellung

- (d) Nennen Sie drei Klassen für das obige Szenario, die direkt durch Basisdatentypen wie Integer oder String abgedeckt werden können.

Lösungsvorschlag

- Nachricht
- Preis
- Rabatt

- (e) Erstellen Sie ein Sequenzdiagramm für das gegebene System mit folgendem Anwendungsfall: Ein Nutzer bestellt zwei Pizzen und eine Flasche Wein. Beginnen Sie mit der „Auswahl des Gerichts“ bis hin zur „Bestätigung der Bestellung“ sowie „Lieferung an die Haustür“. Das Diagramm soll mindestens je einen Akteur für Benutzer (Browser), Applikation (Webserver) und Restaurant (Koch und Liefersdienst) vorsehen. Die Bezahlung selbst muss nicht modelliert werden.

## 66116 / 2020 / Herbst / Thema 2 / Teilaufgabe 1 / Aufgabe 5

Wir betrachten Terme über die Rechenarten  $op \in \{+, -, \cdot, \div\}$ , die rekursiv definiert sind:

- Jedes Literal ist ein Term, z. B. „4“.
- Jedes Symbol ist ein Term, z. B. „x“.
- Ist  $t$  ein Term, so ist „(t)“ ein (geklammerter) Term.
- Sind  $t_1, t_2$  Terme, so ist „ $t_1 op t_2$ “ ebenso ein Term.

Beispiele für gültige Terme sind also „ $4 + 8$ “, „ $4 \cdot x$ “ oder „ $4 + (8 \cdot x)$ “.

- (a) Welches Design-Pattern eignet sich hier am besten zur Modellierung dieses Sachverhalts?

Lösungsvorschlag

Kompositum

- (b) Nennen Sie drei wesentliche Vorteile von Design-Pattern im Allgemeinen.
- (c) Modellieren Sie eine Klassenstruktur in UML, die diese rekursive Struktur von Termen abbildet. Sehen Sie mindestens einzelne Klassen für die *Addition* und *Multiplikation* vor, sowie weitere Klassen für *geklammerte Terme* und *Literale*, welche ganze Zahlen repräsentieren. Gehen Sie bei der Modellierung der Klassenstruktur davon aus, dass eine objektorientierte Programmiersprache wie Java zu benutzen ist.
- (d) Erstellen Sie ein Objektdiagramm, welches den Term  $t := 4 + (3 \cdot 2) + (12 \cdot y / (8 \cdot x))$  entsprechend Ihres Klassendiagramms repräsentiert.
- (e) Überprüfen Sie, ob das Objektdiagramm für den in Teilaufgabe d) gegebenen Term eindeutig definiert ist. Begründen Sie Ihre Entscheidung.
- (f) Die gegebene Klassenstruktur soll mindestens folgende Operationen unterstützen:
- das Auswerten von Termen,
  - das Ausgeben in einer leserlichen Form,

- das Auflisten aller verwendeten Symbole. Welches Design-Pattern ist hierfür am besten geeignet?
- (g) Erweitern Sie Ihre Klassenstruktur um die entsprechenden Methoden, Klassen und Assoziationen, um die in Teilaufgabe f) genannten zusätzlichen Operationen gemäß dem von Ihnen genannten Design Pattern zu unterstützen.

```
public class Addition extends Rechenart {  
    public Addition(Term a, Term b) {  
        super(a, b, "+");  
    }  
  
    public double auswerten() {  
        return a.auswerten() + b.auswerten();  
    }  
}
```

Code-Beispiel auf Github ansehen: [src/main/java/org/bschlangaul/examen/examen\\_66116/jahr\\_2020/herbst/rechenarten/Addition.java](https://github.com/orgs/bschlangaul/repositories/examen/examen_66116/jahr_2020/herbst/rechenarten/Addition.java)

```
public class Divison extends Rechenart {  
    public Divison(Term a, Term b ) {  
        super(a, b, "/");  
    }  
  
    public double auswerten() {  
        return a.auswerten() / b.auswerten();  
    }  
}
```

Code-Beispiel auf Github ansehen: [src/main/java/org/bschlangaul/examen/examen\\_66116/jahr\\_2020/herbst/rechenarten/Divison.java](https://github.com/orgs/bschlangaul/repositories/examen/examen_66116/jahr_2020/herbst/rechenarten/Divison.java)

```
public class GeklammerterTerm extends Term {  
    Term term;  
  
    public GeklammerterTerm(Term term) {  
        this.term = term;  
    }  
  
    public double auswerten() {  
        return term.auswerten();  
    }  
  
    public void ausgeben() {  
        System.out.print("(");  
        term.ausgeben();  
        System.out.print(")");  
    }  
}
```

Code-Beispiel auf Github ansehen: [src/main/java/org/bschlangaul/examen/examen\\_66116/jahr\\_2020/herbst/rechenarten/GeklammerterTerm.java](https://github.com/orgs/bschlangaul/repositories/examen/examen_66116/jahr_2020/herbst/rechenarten/GeklammerterTerm.java)

```
public class Literal extends Term {  
    int wert;
```

```
public Literal(int wert) {
    this.wert = wert;
}

public double auswerten() {
    return wert;
}

public void ausgeben() {
    System.out.print(wert);
}
}
```

Code-Beispiel auf Github ansehen: [src/main/java/org/bschlangaul/examen/examen\\_66116/jahr\\_2020/herbst/rechenarten/Literal.java](https://github.com/bschlangaul/examen/examen_66116/jahr_2020/herbst/rechenarten/Literal.java)

```
public class Multiplikation extends Rechenart {
    public Multiplikation(Term a, Term b) {
        super(a, b, "*");
    }

    public double auswerten() {
        return a.auswerten() * b.auswerten();
    }
}
```

Code-Beispiel auf Github ansehen: [src/main/java/org/bschlangaul/examen/examen\\_66116/jahr\\_2020/herbst/rechenarten/Multiplikation.java](https://github.com/bschlangaul/examen/examen_66116/jahr_2020/herbst/rechenarten/Multiplikation.java)

```
abstract class Rechenart extends Term {
    Term a;
    Term b;
    String operatorZeichen;

    public Rechenart (Term a, Term b, String operatorZeichen) {
        this.a = a;
        this.b = b;
        this.operatorZeichen = operatorZeichen;
    }

    public void ausgeben () {
        a.ausgeben();
        System.out.print(" " + operatorZeichen + " ");
        b.ausgeben();
    }

    abstract public double auswerten();
}
```

Code-Beispiel auf Github ansehen: [src/main/java/org/bschlangaul/examen/examen\\_66116/jahr\\_2020/herbst/rechenarten/Rechenart.java](https://github.com/bschlangaul/examen/examen_66116/jahr_2020/herbst/rechenarten/Rechenart.java)

```
public class Subtraktion extends Rechenart {
    public Subtraktion(Term a, Term b) {
        super(a, b, "-");
    }

    public double auswerten() {
        return a.auswerten() - b.auswerten();
    }
}
```

Code-Beispiel auf Github ansehen: [src/main/java/org/bschlangaul/examen/examen\\_66116/jahr\\_2020/herbst/rechenarten/Subtraktion.java](https://github.com/bschlangaul/examen/examen_66116/jahr_2020/herbst/rechenarten/Subtraktion.java)

```
public class Symbol extends Term {
    String name;
    public Symbol(String name) {
        this.name = name;
    }

    public double auswerten() {
        return Klient.symbole.get(name);
    }

    public void ausgeben() {
        System.out.print(name);
    }
}
```

Code-Beispiel auf Github ansehen: [src/main/java/org/bschlangaul/examen/examen\\_66116/jahr\\_2020/herbst/rechenarten/Symbol.java](https://github.com/bschlangaul/examen/examen_66116/jahr_2020/herbst/rechenarten/Symbol.java)

```
abstract class Term {
    abstract public double auswerten();

    abstract public void ausgeben();
}
```

Code-Beispiel auf Github ansehen: [src/main/java/org/bschlangaul/examen/examen\\_66116/jahr\\_2020/herbst/rechenarten/Term.java](https://github.com/bschlangaul/examen/examen_66116/jahr_2020/herbst/rechenarten/Term.java)

## 66116 / 2020 / Herbst / Thema 2 / Teilaufgabe 1 / Aufgabe 6

Gegeben sei folgendes Java-Programm, das mit der Absicht geschrieben wurde, den größten gemeinsamen Teiler zweier Zahlen zu berechnen:

```
/** Return the Greatest Common Divisor of two integer values. */

public int gcd(int a, int b) {
    if (a < 0 || b < 0) {
        return gcd(-b, a);
    }
    while (a != b) {
        if (a > b) {
            a = a - b;
        } else {
            b = b % a;
        }
    }
    return 3;
}
```

- Bestimmen Sie eine möglichst kleine Menge an Testfällen für das gegebene Programm, um (dennoch) vollständige Zweigüberdeckung zu haben.
- Welche Testfälle sind notwendig, um die Testfälle der Zweigüberdeckung so zu erweitern, dass eine 100% Anweisungsüberdeckung erfüllt ist? Begründen Sie Ihre Entscheidung.

- (c) Beschreiben Sie zwei allgemeine Nachteile der Anweisungsüberdeckung.
- (d) Das gegebene Programm enthält (mindestens) zwei Fehler. Bestimmen Sie jeweils eine Eingabe, die fehlerhaftes Verhalten des Programms verursacht. Nennen Sie den Fault und den Failure, der bei der gewählten Eingabe vorliegt. Sie müssen das Programm (noch) nicht verbessern.
- (e) Geben Sie für die gefundenen zwei Fehler jeweils eine mögliche Verbesserung an. Es reicht eine textuelle Beschreibung, Code ist nicht notwendig.
- (f) Erläutern Sie, warum es im Allgemeinen hilfreich sein kann, bei der Fehlerbehebung in einem größeren Programm die Versionsgeschichte miteinzubeziehen.

## 66116 / 2021 / Frühjahr / Thema 1 / Teilaufgabe 1 / Aufgabe 10

- (a) Was bedeutet die Abkürzung AJAX?

Lösungsvorschlag

Asynchronous JavaScript and XML

- (b) Erklären Sie in max. drei Sätzen die grundlegende Funktion von AJAX.

Lösungsvorschlag

Konzept der asynchronen Datenübertragung zwischen einem Browser und dem Server. Dieses ermöglicht es, HTTP-Anfragen durchzuführen, während eine HTML-Seite angezeigt wird, und die Seite zu verändern, ohne sie komplett neu zu laden.<sup>a</sup>

<sup>a</sup>[https://de.wikipedia.org/wiki/Ajax\\_\(Programmierung\)](https://de.wikipedia.org/wiki/Ajax_(Programmierung))

## 66116 / 2021 / Frühjahr / Thema 1 / Teilaufgabe 1 / Aufgabe 11

- (a) Was ist das Hypertext Transfer Protocol (HTTP) und wozu dient es?

Lösungsvorschlag

Das Hypertext Transfer Protocol ist ein zustandsloses Protokoll zur Übertragung von Daten auf der Anwendungsschicht über ein Rechnernetz. Es wird hauptsächlich eingesetzt, um Webseiten (Hypertext-Dokumente) aus dem World Wide Web (WWW) in einen Webbrowser zu laden. Es ist jedoch nicht prinzipiell darauf beschränkt und auch als allgemeines Dateiübertragungsprotokoll sehr verbreitet.<sup>a</sup>

<sup>a</sup>[https://de.wikipedia.org/wiki/Hypertext\\_Transfer\\_Protocol](https://de.wikipedia.org/wiki/Hypertext_Transfer_Protocol)

- (b) Betrachten Sie die folgende Zeile Text. Um welche Art von Text handelt es sich?

<https://developer.mozilla.org/en-US/search?q=client+servertooverview>

Es handelt sich um eine HTTP-URL (Uniform Resource Locator). Die URL lokalisiert eine Ressource, beispielsweise eine Webseite, über die zu verwendende Zugriffsmethode (zum Beispiel das verwendete Netzwerkprotokoll wie HTTP oder FTP) und den Ort (engl. location) der Ressource in Computernetzwerken.<sup>a</sup>

<sup>a</sup>[https://de.wikipedia.org/wiki/Uniform\\_Resource\\_Locator](https://de.wikipedia.org/wiki/Uniform_Resource_Locator)

- (c) Was sind die vier wesentlichen Bestandteile des Texts aus der vorigen Teilaufgabe?

Lösungsvorschlag

**Schema** https://  
**Host** developer.mozilla.org  
**Pfad** /en-US/search  
**Query** ?q=client+servertooverview

<sup>a</sup>

<sup>a</sup>[https://de.wikipedia.org/wiki/Uniform\\_Resource\\_Locator](https://de.wikipedia.org/wiki/Uniform_Resource_Locator)

## 66116 / 2021 / Frühjahr / Thema 1 / Teilaufgabe 1 / Aufgabe 12

Es gibt Softwaresysteme, welche auf peer-to-peer (P2P) Kommunikation basieren und eine entsprechende Architektur aufweisen.

- (a) Bewerten Sie die folgenden Aussagen als entweder richtig oder falsch.
- (i) Mithilfe des Befehls "lookup" können Peers sich gegenseitig identifizieren.

Lösungsvorschlag

richtig

- (ii) In einem P2P-System, wie auch bei Client-Server, sind alle Netzwerkteilnehmer gleichberechtigt.

Lösungsvorschlag

falsch. Im Client-Servermodell sind nicht alle Netzwerkteilnehmer gleichberechtigt. Der Server hat mehr Privilegien wie der Client.

- (iii) Alle P2P-Systeme funktionieren grundsätzlich ohne einen zentralen Verwaltungs-Peer.

Lösungsvorschlag

falsch. Es gibt zentralisierte P2P-Systeme (Beispiel: Napster), welche einen zentralen Server zur Verwaltung benötigen, um zu funktionieren.



*a*<sup>a</sup><https://de.wikipedia.org/wiki/Peer-to-Peer>

UML-Diagramme

- (iv) P2P kann auch für eine Rechner-Rechner-Verbindung stehen.

Lösungsvorschlag

richtig

- (v) Es gibt strukturierte und unstrukturierte P2P-Systeme. In unstrukturierten P2P-Systemen wird zum Auffinden von Peers eine verteilte Hashtabelle verwendet (DHT).

Lösungsvorschlag

richtig

- (vi) In einem P2P-System sind theoretisch alle Peers gleichberechtigt, praktisch gibt es jedoch leistungsabhängige Gruppierungen.

Lösungsvorschlag

richtig

- (vii) Ein Peer kann sowohl ein Client wie auch ein Server für einen anderen Peer sein.

Lösungsvorschlag

richtig

- (b) Wählen Sie zwei falsche Aussagen aus der vorherigen Tabelle aus und berichtigen Sie diese in jeweils einem Satz.

Lösungsvorschlag

Sie oben.

### 66116 / 2021 / Frühjahr / Thema 1 / Teilaufgabe 1 / Aufgabe 3

Wählen Sie bis zu fünf unterschiedliche Elementtypen aus folgendem Diagramm aus und benennen Sie diese Elemente und ihre syntaktische (nicht semantische) Bedeutung.

Lösungsvorschlag

**Klasse** ConcreteCreator**Interface** Produkt**Abstrakte Klasse** Creator**Kommentar** return new ConcreteProdukt()

## 66116 / 2021 / Frühjahr / Thema 1 / Teilaufgabe 1 / Aufgabe 6

- (a) Erläutern Sie den Zweck (Intent) des Erzeugungsmusters Erbauer in max. drei Sätzen, ohne dabei auf die technische Umsetzung einzugehen.

Lösungsvorschlag

Die Erzeugung komplexer Objekte wird vereinfacht, indem der Konstruktionsprozess in eine spezielle Klasse verlagert wird. Er wird so von der Repräsentation getrennt und kann sehr unterschiedliche Repräsentationen zurückliefern.

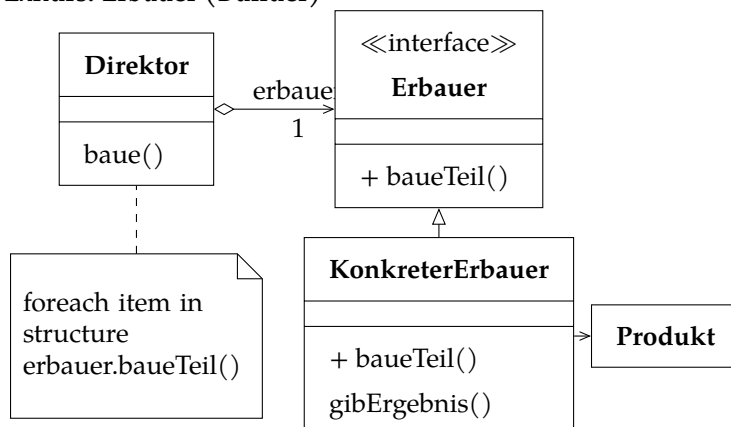
- (b) Erklären Sie, wie das Erzeugungsmuster Erbauer umgesetzt werden kann (Implementierung). Die Angabe von Code ist hierbei NICHT notwendig!
- (c) Nennen Sie jeweils einen Vor- und einen Nachteil des Erzeugungsmusters Erbauer im Vergleich zu einer Implementierung ohne dieses Muster.

Lösungsvorschlag

**Vorteil** Die Implementierungen der Konstruktion und der Repräsentationen werden isoliert. Die Erbauer verstecken ihre interne Repräsentation vor dem Direktor.

**Nachteil** Es besteht eine enge Kopplung zwischen Produkt, konkretem Erbauer und den am Konstruktionsprozess beteiligten Klassen.

## Exkurs: Erbauer (Builder)



**Erbauer** Der Erbauer spezifiziert eine abstrakte Schnittstelle zur Erzeugung der Teile eines komplexen Objektes.

**KonkreterErbauer** Der konkrete Erbauer erzeugt die Teile des komplexen Objekts durch Implementierung der Schnittstelle. Außerdem definiert und verwaltet er die von ihm erzeugte Repräsentation des Produkts. Er bietet auch eine Schnittstelle zum Auslesen des Produkts.

**Direktor** Der Direktor konstruiert ein komplexes Objekt unter Verwendung der Schnittstelle des Erbauers. Der Direktor arbeitet eng mit dem Erbauer zusammen: Er weiß, welche Baureihenfolge der Erbauer verträgt oder benötigt. Der Direktor entkoppelt somit den Konstruktionsablauf vom Klienten.

**Produkt** Das Produkt repräsentiert das zu konstruierende komplexe Objekt.

Einbringen von  
Abhängigkeiten  
(Dependency Injection)  
Client-Server-Modell

## 66116 / 2021 / Frühjahr / Thema 1 / Teilaufgabe 1 / Aufgabe 7

Lesen Sie die folgenden alternativen Codestücke.

- (a)
- ```
public class MyParser {
    private InputStream input;

    public MyParser(String filePath) {
    }
}
```
- (b)
- ```
public class MyParser {
    private InputStream input;

    public MyParser(InputStream stream) {
    }
}
```

Beide Codestücke zeigen die Initialisierung einer Klasse namens `MyParser`. Das zweite Codestück nutzt jedoch hierfür eine Technik namens Abhängigkeits-Injektion (Dependency Injection).

- (a) Erklären Sie den Unterschied zwischen beiden Initialisierungen. Hinweis: Sie können diese Aufgabe auch lösen, falls Sie die Technik nicht kennen.

Lösungsvorschlag

Die Abhängigkeit von einer Instanz der Klasse `InputStream` wird erst bei der Initialisierung des Objekt übergeben.

- (b) Benennen Sie einen Vorteil dieser Technik.

Lösungsvorschlag

Die Kopplung zwischen einer Klasse und ihrer Abhängigkeit wird verringert.

## 66116 / 2021 / Frühjahr / Thema 1 / Teilaufgabe 1 / Aufgabe 8

Das Client-Server-Modell ist ein Architekturmuster. Nennen Sie zwei Vorteile einer nach diesem Muster gestalteten Architektur.

Lösungsvorschlag

- (a) Einfache Integration weiterer Clients
- (b) Prinzipiell uneingeschränkte Anzahl der Clients <sup>a</sup>
- (c) Es muss nur ein Server gewartet werden. Dies gilt z. B. für Updates, die einmalig und zentral auf dem Server durchgeführt werden und danach für alle

Clients verfügbar sind.<sup>b</sup>

Client-Server-Modell  
CPM-Netzplantechnik

<sup>a</sup><https://www.karteikarte.com/card/164928/vorteile-und-nachteile-des-client-server-modells>

<sup>b</sup><https://www.eoda.de/wissen/blog/client-server-architekturen-performance-und-agilitaet-fuer-data-s>

## 66116 / 2021 / Frühjahr / Thema 1 / Teilaufgabe 1 / Aufgabe 9

Betrachten Sie die folgende Liste von Technologien:

- Nodejs
- PHP
- CSS
- AJAX
- Python
- Java

Welche dieser Technologien laufen in einem Client-Server-System üblicherweise auf der Seite des Klienten und welche auf der Seite des Servers? Nehmen Sie hierzu an, dass der Client ein Browser ist.

Übertragen Sie die im Folgenden gegebene Tabelle in Ihren Bearbeitungsbogen und ordnen Sie die aufgelisteten Technologien anhand der Buchstaben in die Tabelle ein. Fortsetzung nächste Seite!

Hinweis: Mehrfachzuordnungen sind möglich.

Lösungsvorschlag

Client-seitige Technologien	Server-seitige Technologien
CSS	Nodejs
AJAX	PHP
Python	Python
Java	Java

## 66116 / 2021 / Frühjahr / Thema 2 / Teilaufgabe 1 / Aufgabe 1

Gegeben seien folgende Tätigkeiten mit ihren Abhängigkeiten und Dauern:

Task	Dauer (in h)	Abhängigkeiten
T1	3	/
T2	6	/
T3	2	T1
T4	2	T2
T5	5	T1
T6	3	T4, T5
T7	6	T3
T8	7	T4
T9	4	T6, T8
T10	1	T7, T9

- (a) Zeichnen Sie ein CPM-Diagramm basierend auf der gegebenen Aufgabenliste. Benutzen Sie explizite Start- und Endknoten.

Lösungsvorschlag

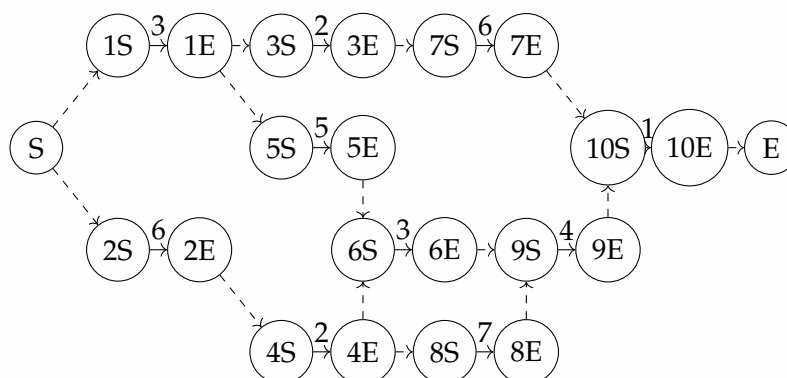
### Abkürzungen

**S** Start

**1S** Start von T1

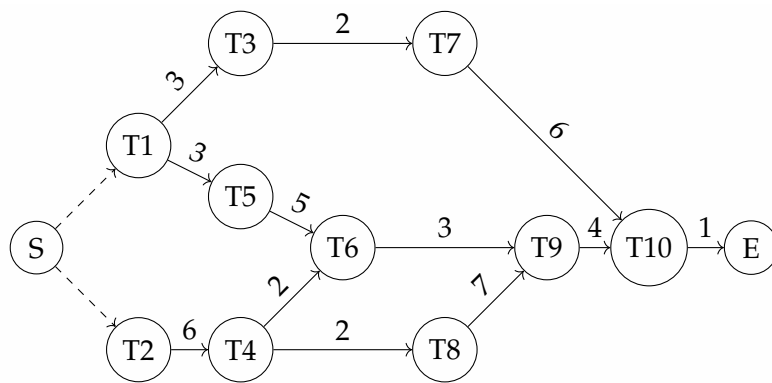
**1E** Ende von T1

**E** Ende

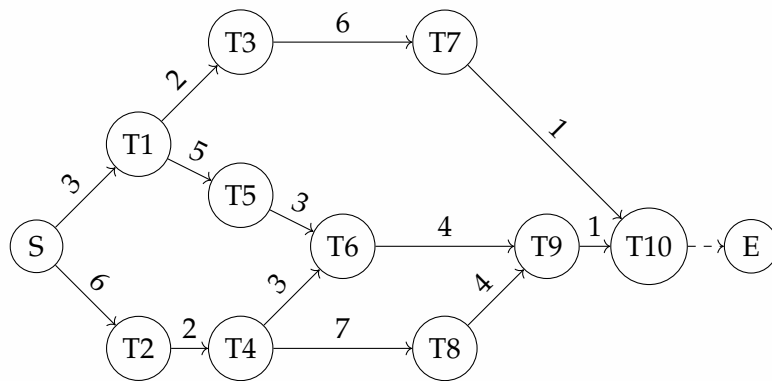


Teilen wir einen Task in zwei Knoten auf, so wird das Diagramm sehr unübersichtlich. Wir verwenden pro Task nur einen Knoten. Es gibt zwei Möglichkeiten:

**Knoten sind Anfang der Tasks**



### Knoten sind Ende der Tasks



- (b) Als *Slack* bezeichnet man die Zeit, um die eine Aufgabe bezüglich ihres frühesten Startzeitpunktes verzögert werden kann, ohne dass es Probleme bei der fristgerechten Fertigstellung des Projektes gibt. Berechnen Sie den Slack für alle Aktivitäten und ergänzen Sie ihn in Ihrem Diagramm.

Lösungsvorschlag

### Knoten sind Anfang der Tasks

— Wir führen eine Vorwärtsterminierung durch und addieren die Dauern. Kann ein Ereignis über mehrere Vorgänge erreicht werden, wählen wir das Maximum aus. **Erläuterungen:**  $i$ : Ereignis  $i$ ;  $FZ_i$ : Frühester Zeitpunkt, zu dem Ereignis  $i$  eintreten kann. \_\_\_\_\_

$i$	Nebenrechnung	$FZ_i$
T1		0
T2		0
T3		3
T4		6
T5		3
T6	$\max(8_{T4}, 8_{T5})$	8
T7		5
T8		8
T9	$\max(11_{T6}, 15_{T4})$	15
T10	$\max(19_{T9}, 11_{T7})$	19
E		20

— Wir führen eine Rückwärtsterminierung durch und subtrahieren die Dauern vom letzten Ereignis aus. Kann ein Ereignis über mehrere Vorgänge erreicht werden, wählen wir das Minimum aus. **Erläuterungen:**  $i$ : Ereignis  $i$ ;  $SZ_i$ : Spätester Zeitpunkt, zu dem Ereignis  $i$  eintreten kann. —

$i$	Nebenrechnung	$SZ_i$
E		20
T10		19
T9		15
T8		8
T7		13
T6		12
T5		7
T4	$\min(12_{T6}, 6_{T8})$	6
T3		11
T2		0
T1	$\min(8_{T3}, 4_{T5})$	4

$i$	T1	T2	T3	T4	T5	T6	T7	T8	T9	T10	E
$FZ_i$	0	0	3	6	3	8	5	8	15	19	20
$SZ_i$	4	0	11	6	7	12	13	8	15	19	20
GP	4	0	8	0	4	4	8	0	0	0	0

**Knoten sind Ende der Tasks**

— Wir führen eine Vorwärtsterminierung durch und addieren die Dauern. Kann ein Ereignis über mehrere Vorgänge erreicht werden, wählen wir das Maximum aus. **Erläuterungen:**  $i$ : Ereignis  $i$ ;  $FZ_i$ : Frühester Zeitpunkt, zu dem Ereignis  $i$  eintreten kann.

$i$	Nebenrechnung	$FZ_i$
T1		3
T2		6
T3		5
T4		8
T5		8
T6	$\max(11_{T4}, 11_{T5})$	11
T7		11
T8		15
T9	$\max(15_{T6}, 19_{T8})$	19
T10	$\max(20_{T9}, 12_{T7})$	20
E		20

— Wir führen eine Rückwärtsterminierung durch und subtrahieren die Dauern vom letzten Ereignis aus. Kann ein Ereignis über mehrere Vorgänge erreicht werden, wählen wir das Minimum aus. **Erläuterungen:**  $i$ : Ereignis  $i$ ;  $SZ_i$ : Spätester Zeitpunkt, zu dem Ereignis  $i$  eintreten kann.

$i$	Nebenrechnung	$SZ_i$
E		20
T10		20
T9		19
T8		15
T7		19
T6		15
T5		12
T4	$\min(12_{T6}, 8_{T8})$	8
T3		13
T2		6
T1	$\min(11_{T3}, 7_{T5})$	7



$i$	T1	T2	T3	T4	T5	T6	T7	T8	T9	T10	E
$FZ_i$	3	6	5	8	8	11	11	15	19	20	20
$SZ_i$	7	6	13	8	12	15	19	15	19	20	20
GP	4	0	8	0	4	4	8	0	0	0	0

- (c) Zeichnen Sie den kritischen Pfad in Ihr Diagramm ein oder geben Sie die Tasks des kritischen Pfades in der folgenden Form an: **Start ! ... ! Ende**. Sollte es mehrere kritische Pfade geben, geben Sie auch diese an. Wie lange ist die Dauer des kritischen Pfades bzw. der kritischen Pfade?

Lösungsvorschlag

Kritischer Pfad: **Start ! T2 ! T4 ! T8 ! T9 ! T10 ! Ende**  
Dauer: 20 h

## 66116 / 2021 / Frühjahr / Thema 2 / Teilaufgabe 1 / Aufgabe 2

- (a) Erklären Sie den Unterschied zwischen *iterativen* und *inkrementellen* Entwicklungsprozessen. Nennen Sie zudem je ein Prozessmodell als Beispiel.

Lösungsvorschlag

**Iterativ:** Ein Entwicklungszyklus wird immer wieder durchlaufen:

Planung → Implementierung → Testung → Evaluation.

Mit jeder Iteration wird das Produkt verfeinert. Das Wasserfallmodell in seiner normalen Form würde dies nicht erfüllen, da hier alle Phasen nur einmal durchlaufen werden.

**Beispiel:** Agile Programmierung, erweitertes Wasserfallmodell

**Inkrementell:** Das Projekt wird in einzelne Teile zerlegt. An jedem Teilprojekt kann bestenfalls separat gearbeitet werden. In den einzelnen Teilprojekten kann dann ebenfalls wieder iterativ gearbeitet werden, die beiden Methoden schließen sich gegenseitig also nicht aus.

**Beispiel:** Agile Programmierung, V-Modell XT, Aufteilung in Teilprojekte, die alle mit Wasserfallmodell bearbeitet werden

- (b) Nennen und erklären Sie kurz die vier Leitsätze der agilen Softwareentwicklung laut *Agilem Manifest*.

(i) **Individuen und Interaktionen sind wichtiger als Prozesse und Werkzeuge**

Ein festgelegter Prozess, der nicht sinnvoll von den beteiligten Personen umgesetzt werden kann, ist nicht sinnvoll und sollte nicht durchgeführt

werden. Es geht darum, die Stärken der Mitarbeiter einzusetzen und sich nicht sklavisches an Abläufen zu orientieren. Es geht darum, Freiräume zu schaffen, um das Potential des Einzelnen voll entfalten zu lassen.

(ii) **Funktionierende Software ist wichtiger als umfassende Dokumentation**

Der Kunde ist in erster Linie an einem funktionierenden Produkt interessiert. Es soll möglichst früh ein Prototyp entstehen, der dann im weiteren Prozess an die Bedürfnisse des Kunden angepasst wird. Eine umfassende Dokumentation kann am Ende immer noch ergänzt werden.

(iii) **Zusammenarbeit mit dem Kunden ist wichtiger als Vertragsverhandlung**

Anforderungen und Spezifikationen können sich flexibel ändern, dadurch kann der Kunde direkt Rückmeldung geben, was ist nicht wichtig im Vertrag kleinste Details zu regeln, sondern auf die Bedürfnisse des Kunden einzugehen. Daraus folgt direkt der letzte Punkt:

(iv) **Reagieren auf Veränderung ist wichtiger als das Befolgen eines Plans**

Softwareentwicklung ist ein dynamischer Prozess. Das starre Befolgen eines Plans widerspricht der Grundidee der agilen Programmierung.

- (c) Beschreiben Sie die wesentlichen Aktivitäten in *Scrum* (agiles Entwicklungsmodell) inklusive deren zeitlichen Ablaufs. Gehen Sie dabei auch auf die *Artefakte* ein, die im Verlauf der Entwicklung erstellt werden.

Lösungsvorschlag

(i) **Initiale Projekterstellung:**

Festlegen eines Product Backlog durch den Product Owner, Erstellen von ersten User Stories. Diese entstehen gegebenenfalls durch den Kontakt mit den Stakeholdern, falls der Product Owner nicht selbst der Kunde ist.

(ii) **Sprint planning:**

Der nächste Sprint (zwischen 1 und 4 Wochen) wird geplant. Dabei wird ein Teil des Product Backlog als *Sprint Backlog* definiert und auf die einzelnen Entwickler verteilt. Es wird festgelegt, *was* implementiert werden soll (Product Owner anwesend) und *wie* das geschehen soll (Entwicklerteam). Während des Sprints findet jeden Tag ein *Daily Scrum* statt, ein fünfzehnminütiger Austausch zum aktuellen Stand. Am Ende des Sprints gibt es ein *Sprint Review* und das *Product Inkrement* wird evaluiert und das *Product Backlog* gegebenenfalls angepasst in Absprache mit *Product Owner* und *Stakeholdern*.

**Artefakte:**

- Product Backlog
- Sprint Backlog

- (d) Nennen und erklären Sie die *Rollen* in einem Scrum-Team.

Lösungsvorschlag

**Product Owner** Verantwortlich für das Projekt, für die Reihenfolge der Bearbeitung und Implementierung, ausgerichtet auf Maximierung und wirtschaftlichen Erfolg. Er führt und aktualisiert das *Product Backlog*.

**Scrum Master** Führungsperson, die das Umsetzen des Scrum an sich leitet und begleitet. Er mischt sich nicht mit konkreten Arbeitsanweisungen ein, sondern moderiert und versucht Hindernisse zu beseitigen, die einem Gelingen der Sprintziele im Weg sind.

**Entwickler** Sie entwickeln und implementieren die einzelnen Produktfunktionalitäten, die vom Product Owner festgelegt wurden, in eigenverantwortlicher Zeit. Sie müssen die Qualitätsstandards beachten.

## 66116 / 2021 / Frühjahr / Thema 2 / Teilaufgabe 1 / Aufgabe 4

- (a) Nennen und erklären Sie kurz die drei Kategorien der Organisation von klassischen Entwurfsmustern. Geben Sie zu jeder Kategorie ein Beispiel-Pattern an.
- (b) Erstellen Sie ein Klassendiagramm zu den Komponenten einer Beobachtungsstation, welche aus einem einzigen Wasserstandsmesser besteht. Dabei sollen bei Änderungen des Wasserstandes der aktuelle Wasserstand sowohl auf der Konsole als auch in eine Log-Datei geschrieben werden. Der momentane Wasserstand soll dabei mittels einer Variablen des Datentyps `double` dargestellt werden. Die verschiedenen Anzeigearten (Konsolenanzeige, Logger) sollen als verschiedene Klassen modelliert werden und enthalten jeweils nur eine Methode zum Anzeigen bzw. Schreiben des aktuellen Wasserstandes. Verwenden Sie für die Realisierung dieses Klassendiagramms die passenden Entwurfsmuster.

Hinweise:

- Getter und Setter müssen nicht eingezeichnet werden.
- Fügen Sie auch Methoden ein, welche durch die einzelnen Klassen implementiert werden.