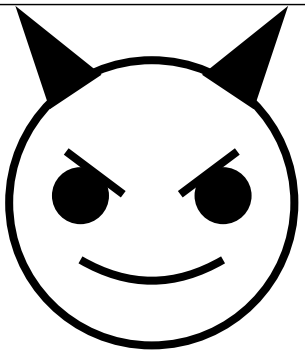
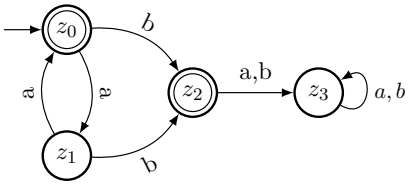
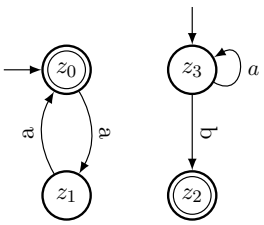
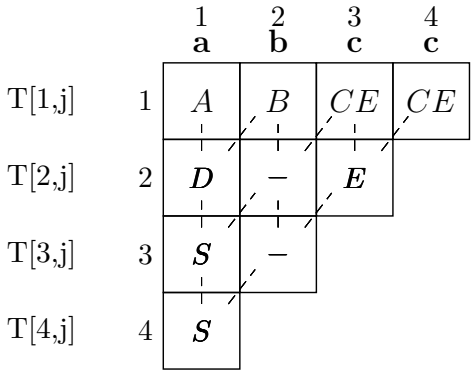
Inhaltsverzeichnis

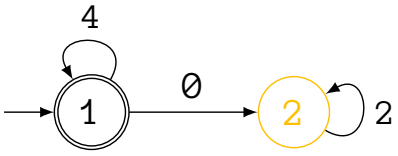
```
%%Einbindung erfolgt über:
\getGraphics{⟨Pfad⟩}
```

Pfad	**Ergebnis**
`Allerlei/Teufel`	
`Automat/AutomatDFA`	Deterministic Finite Automaton  $D = \left(\underset{Menge}{Z}, \underset{Alphabet}{\Sigma}, \underset{Z\times\Sigma\to Z}{\delta}, \underset{\in Z}{S}, \underset{\subseteq Z}{E}\right)$ (Zustände, Eingabe, Übergangsfkt., Start(s), Endzust.) $T(D) = \left\{x \in \Sigma^* \mid \hat{\delta}(S,x) \cap E \neq \varnothing\right\}$
`Automat/AutomatNFA`	Nondeterministic Finite Automaton  $N = \left(\underset{Menge}{Z}, \underset{Alphabet}{\Sigma}, \underset{Z\times\Sigma\to Z^*}{\delta}, \underset{\subseteq Z}{S}, \underset{\subseteq Z}{E}\right)$ (Zustände, Eingabe, Übergangsfkt., Start(s), Endzust.) $T(N) = \left\{x \in \Sigma^* \mid \hat{\delta}(S,x) \cap E \neq \varnothing\right\}$

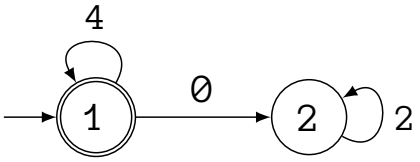
Automat/CYKAlgorithmus



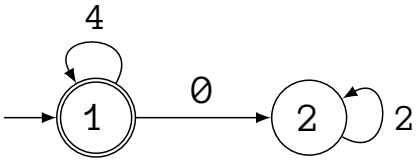
Automat/Demo-1



Automat/Demo-2



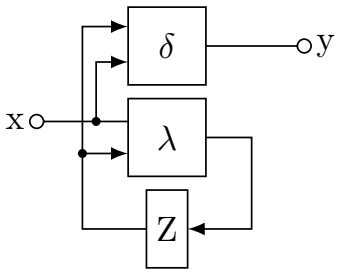
Automat/Demo



Automat/Header

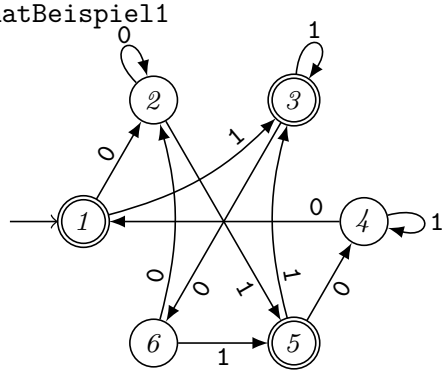
Isch bin a Hädder!

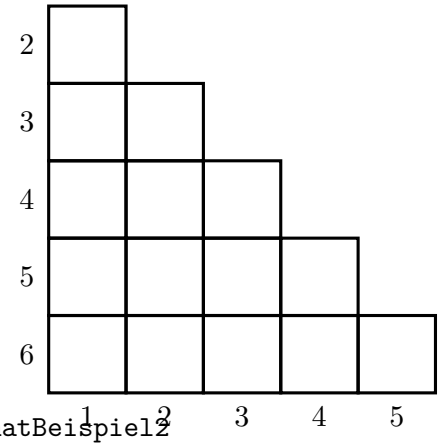
Automat/MealyAutomat



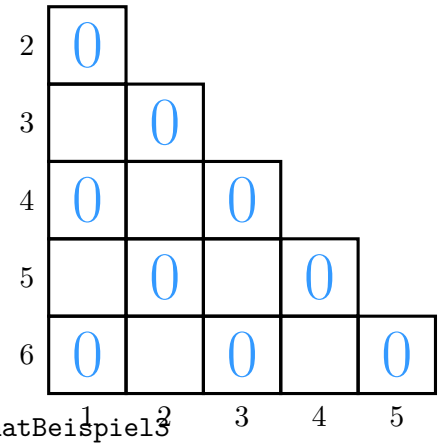
Ausgabe von Zustand & Eingabe abhängig

Automat/Minimalautomat Beispiel/MinimalautomatBeispiel1

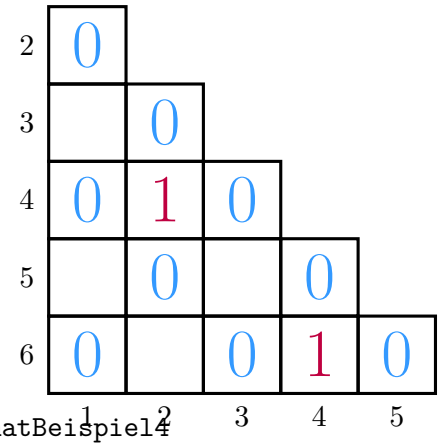




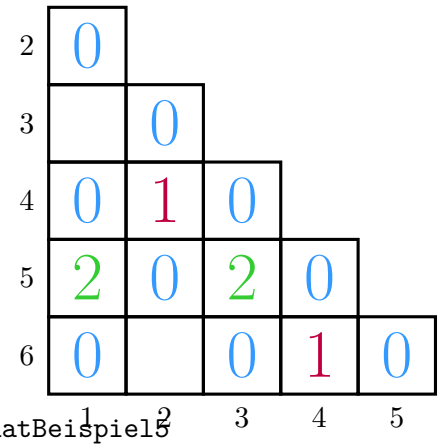
Automat/Minimalautomat Beispiel/MinimalautomatBeispiel2



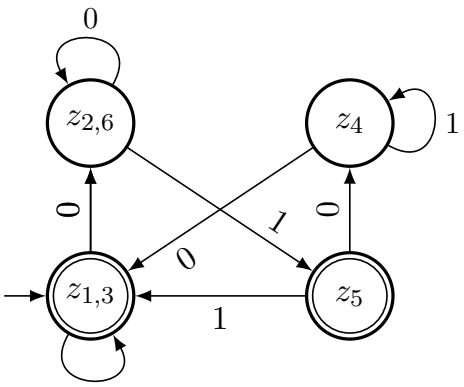
Automat/Minimalautomat Beispiel/MinimalautomatBeispiel3



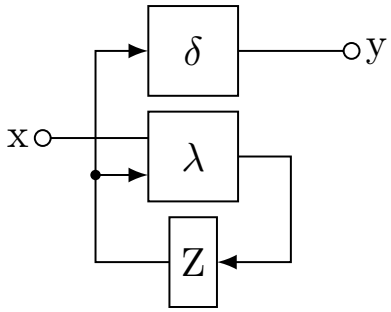
`Automat/MinimalautomatBeispiel/MinimalautomatBeispiel4`



`Automat/MinimalautomatBeispiel/MinimalautomatBeispiel5`

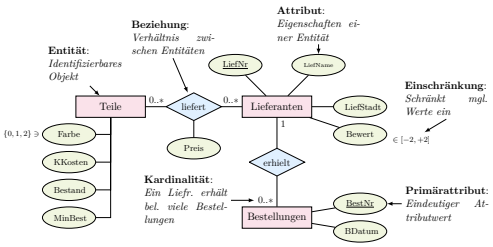


`Automat/MinimalautomatBeispiel/MinimalautomatBeispiel6`

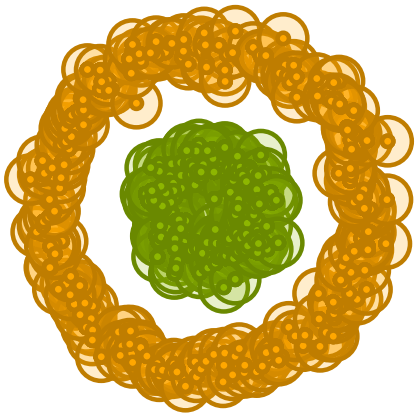


Automat/MooreAutomat

Ausgabe nur vom Zustand abhängig

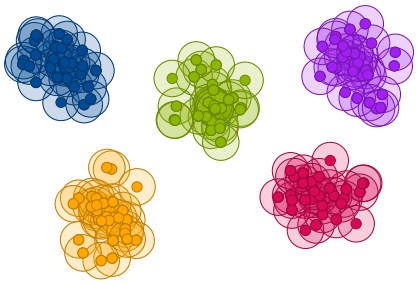


Datenbanken/ERMExample



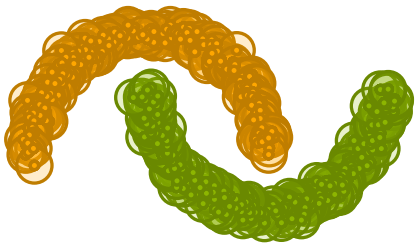
Eigene/Proseminar/Cluster/en-circles

pdf



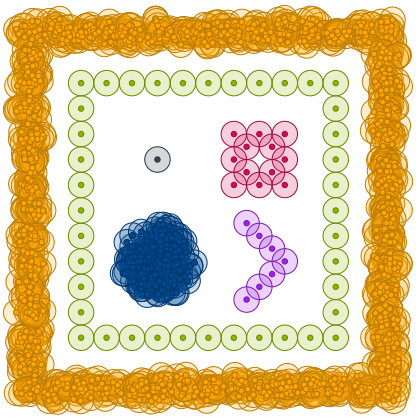
Eigene/Proseminar/Cluster/en-clusters

pdf



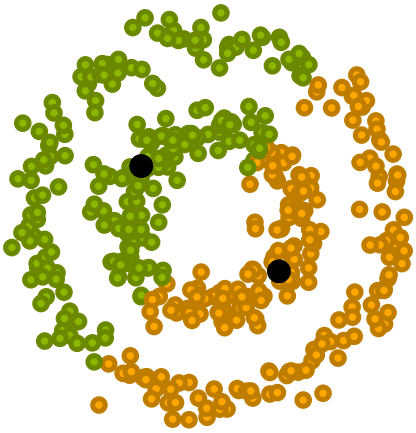
Eigene/Proseminar/Cluster/en-moons

pdf



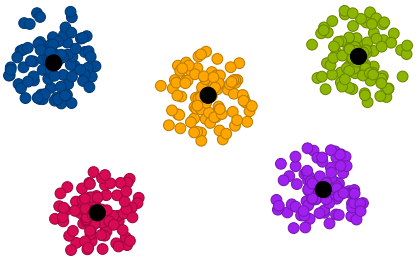
Eigene/Proseminar/Cluster/en-special

pdf



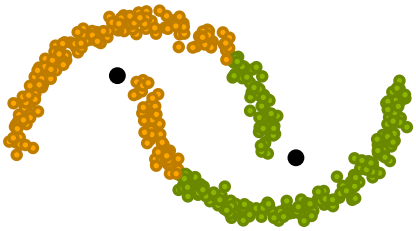
Eigene/Proseminar/Cluster/km-circles

pdf



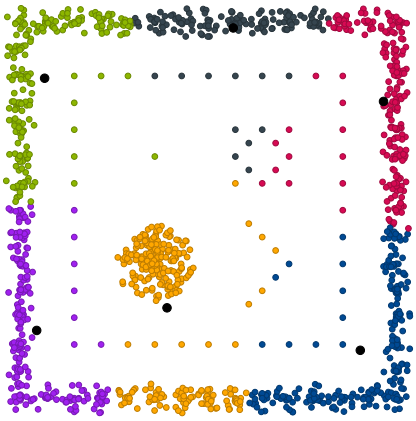
Eigene/Proseminar/Cluster/km-clusters

pdf



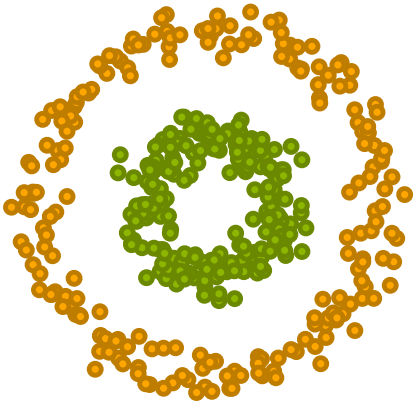
Eigene/Proseminar/Cluster/km-moons

pdf



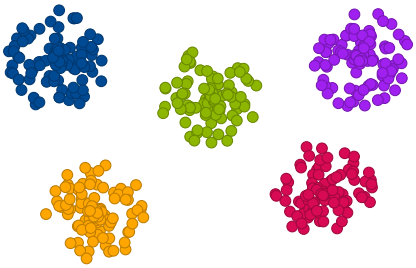
Eigene/Proseminar/Cluster/km-special

pdf



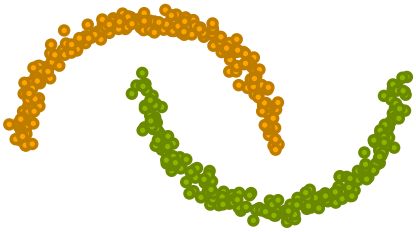
Eigene/Proseminar/Cluster/kn-circles

pdf



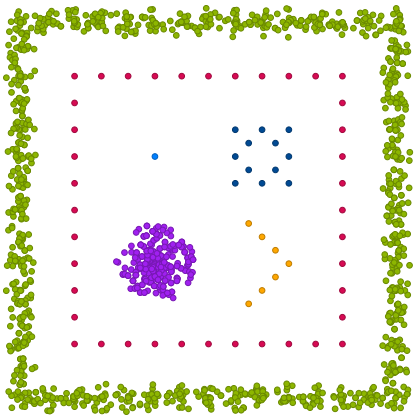
Eigene/Proseminar/Cluster/kn-clusters

pdf



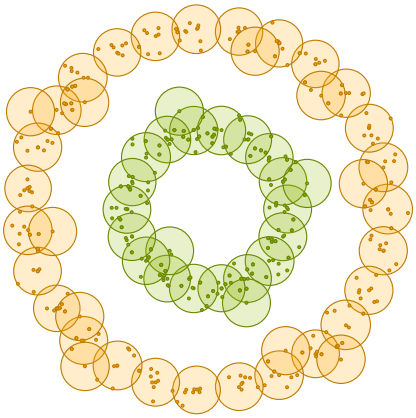
Eigene/Proseminar/Cluster/kn-moons

pdf



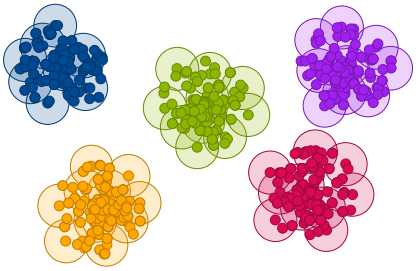
Eigene/Proseminar/Cluster/kn-special

pdf



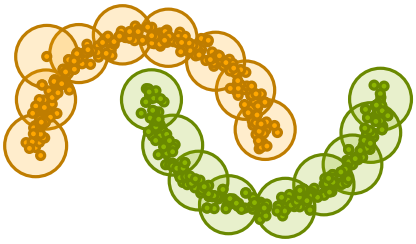
Eigene/Proseminar/Cluster/rolf-circles

pdf



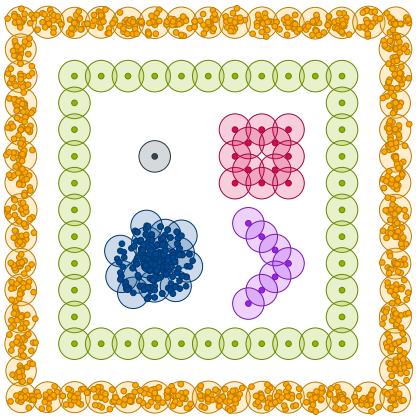
Eigene/Proseminar/Cluster/rolf-clusters

pdf



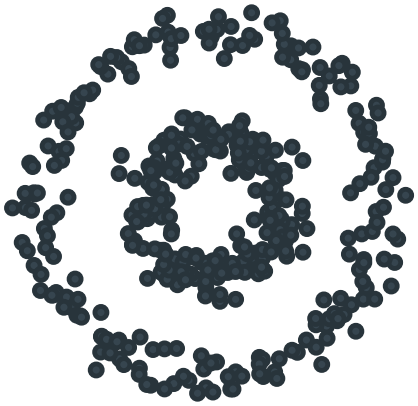
Eigene/Proseminar/Cluster/rolf-moons

pdf



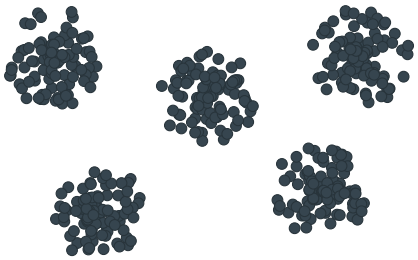
Eigene/Proseminar/Cluster/rolf-special

pdf



Eigene/Proseminar/Cluster/thumb-circles

pdf

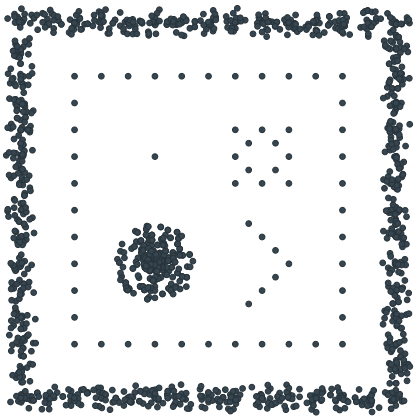


Eigene/Proseminar/Cluster/thumb-clusters

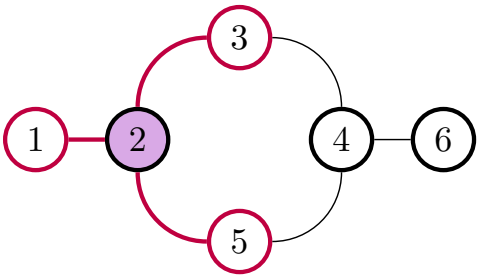
pdf

Eigene/Proseminar/Cluster/thumb-moons

pdf

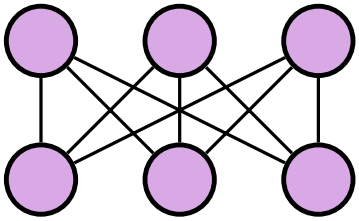


Eigene/Proseminar/Cluster/thumb-special
pdf



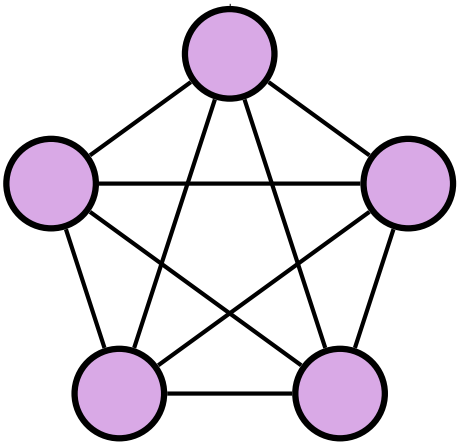
Graphen/GraphNachbarschaftGrad

$K_{3,3}$

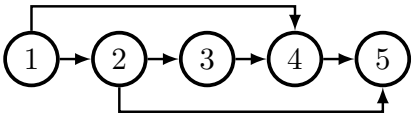


Graphen/GraphNichtPlanarK33

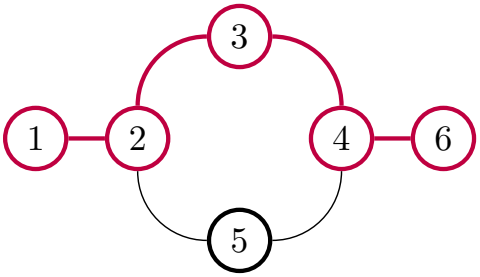
K_5



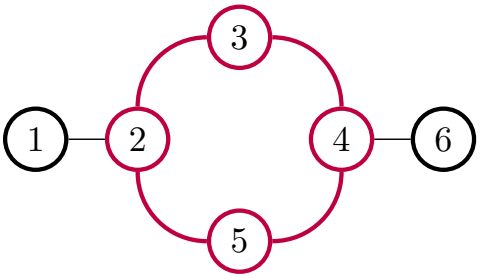
Graphen/GraphNichtPlanarK5



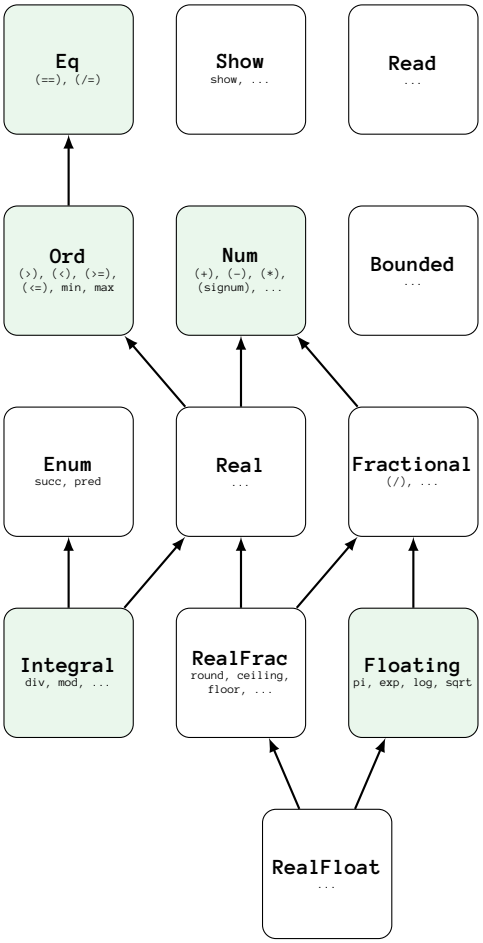
Graphen/GraphTopologie



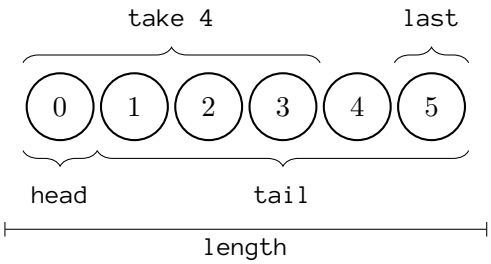
Graphen/GraphWegPfad



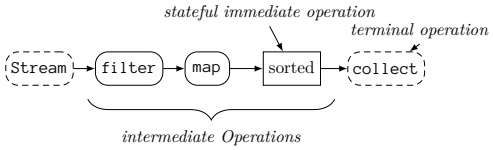
Graphen/GraphZyklus



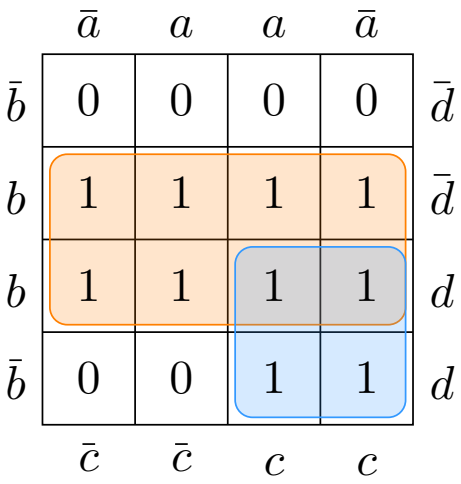
Haskell/HaskellTypen



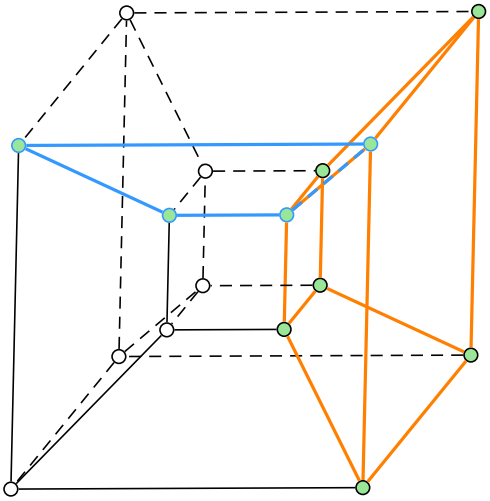
Haskell/Listenoperationen



Java/StreamDemo



Logik/KVDiagramm

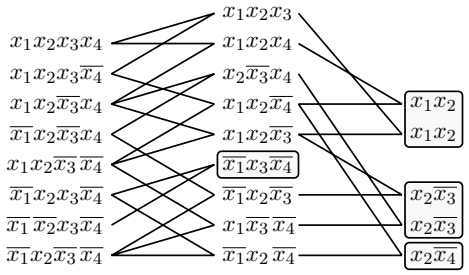


Logik/KVWuerfel

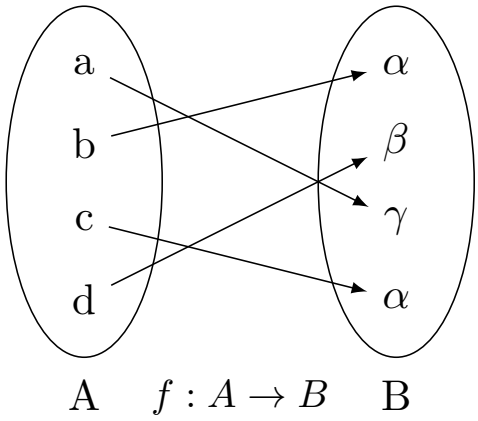
Logik/QuineMCCluskeyTabelle

	1	2	3	4	5	6	7	8
x_1x_2	+	+	+		+			
$x_2\overline{x_3}$			+	+	+			+
$x_2\overline{x_4}$		+			+	+		+
$\overline{x_1}x_3\overline{x_4}$						+	+	

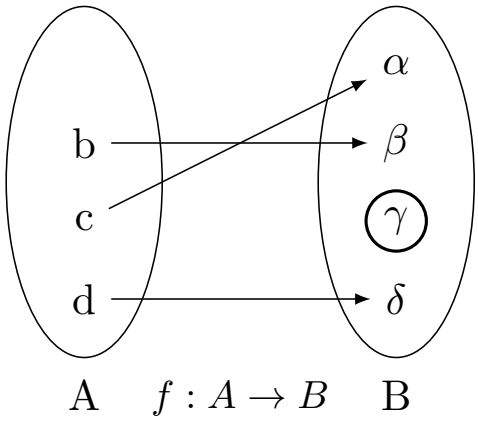
Logik/QuineMCCluskeyZusammenfassen

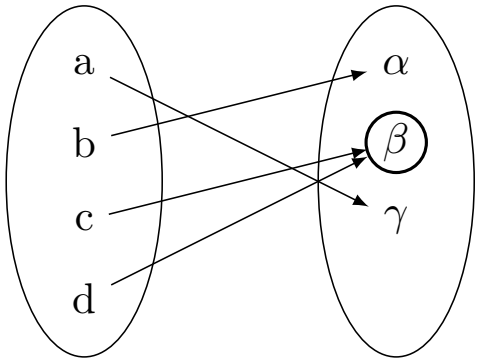


Mengen/FunktionBijektiv

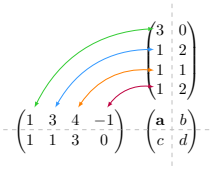


Mengen/FunktionInjektiv



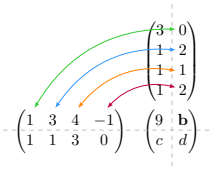


Mengen/FunktionSurjektiv



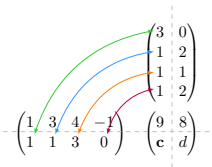
Für a ergibt sich also:
$a = 1 * 3 + 3 * 1 + 4 * 1 + (-1) * 1$
$= 9$

Mengen/Mengenmultiplikation/Mengenmultiplikation1



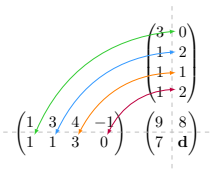
Für b ergibt sich also:
$b = 1 * 0 + 3 * 2 + 4 * 1 + (-1) * 2$
$= 8$

Mengen/Mengenmultiplikation/Mengenmultiplikation2



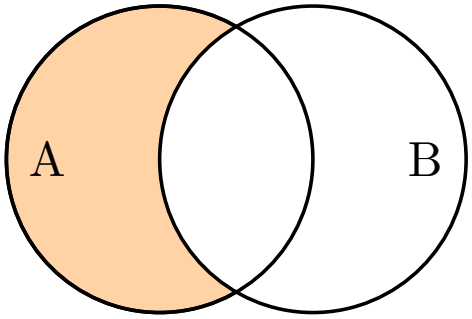
Für c ergibt sich also:
$c = 1 * 3 + 1 * 1 + 3 * 1 + 0 * 1$
$= 7$

Mengen/Mengenmultiplikation/Mengenmultiplikation3

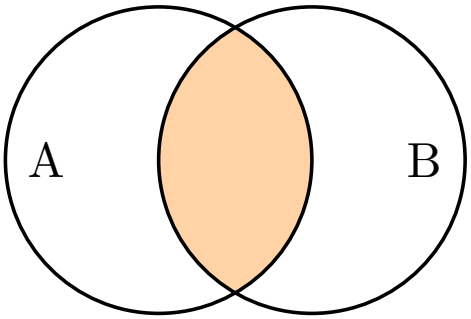


Für d ergibt sich also:
$d = 1 * 0 + 1 * 2 + 3 * 1 + 0 * 2$
$= 5$

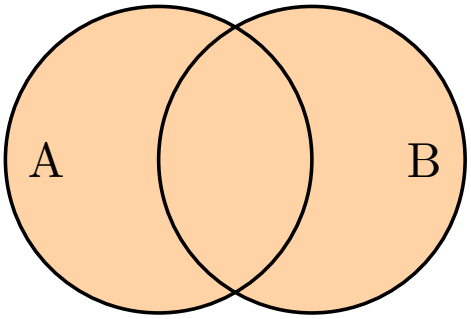
Mengen/Mengenmultiplikation/Mengenmultiplikation4



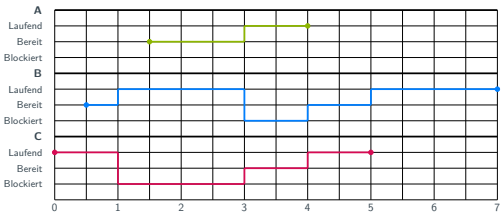
Mengen/VennDifferenz



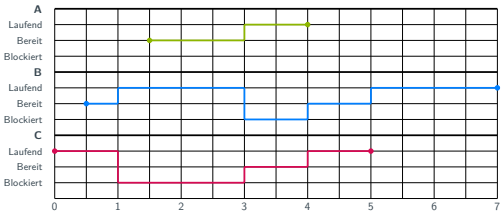
Mengen/VennSchnitt



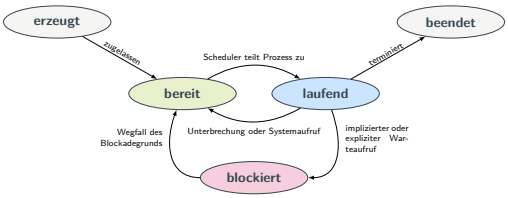
Mengen/VennVereinigung



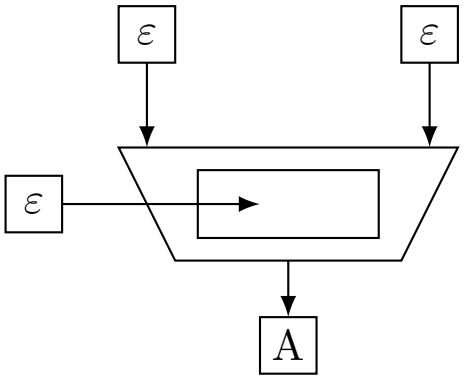
Prozesse/FCFS-WorstCase



Prozesse/FCFS



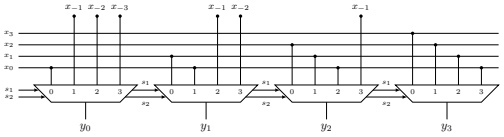
Prozesse/Prozesszustaende



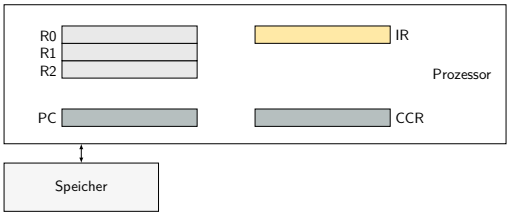
Rechner/ALU

	Zustand			Folgez.		NS			WO			
δ	h_0	z_0	z_1	z_0'	z_1'							T
δ_0	0	0	0	0	1	●	○	○	○	○	●	$7 \Rightarrow h_0' = 1$
δ_1	1	0	1	0	1	●	○	○	○	○	●	$T - 1$
δ_3	0	0	1	1	0	○	●	○	○	●	○	$2 \Rightarrow h_0' = 1$
δ_4	1	1	0	1	0	○	●	○	○	●	○	$T - 1$
δ_6	0	1	0	1	1	○	○	●	●	○	○	$7 \Rightarrow h_0' = 1$
δ_7	1	1	1	1	1	○	○	●	●	○	○	$T - 1$
δ_9	0	1	1	0	0	○	●	○	○	●	○	$2 \Rightarrow h_0' = 1$
δ_{10}	1	0	0	0	0	○	●	○	○	●	○	$T - 1$

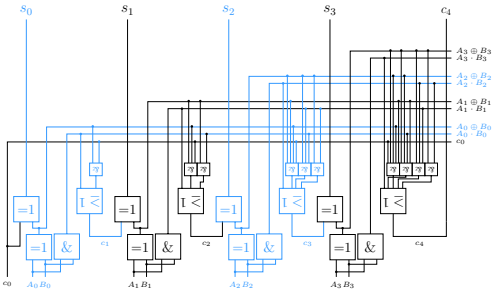
Rechner/AmpelPLA



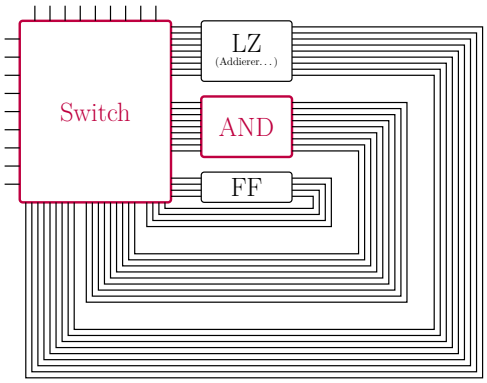
Rechner/BarrelShifter



Rechner/Beispielprozessor

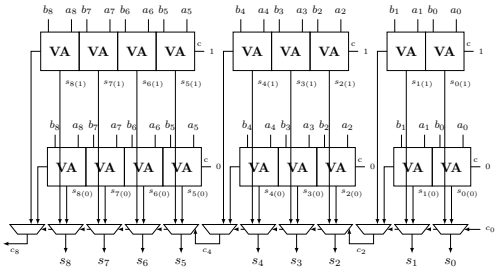


Rechner/CLA

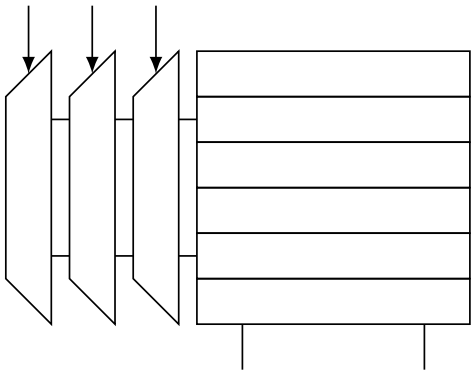


Rechner/CPLD

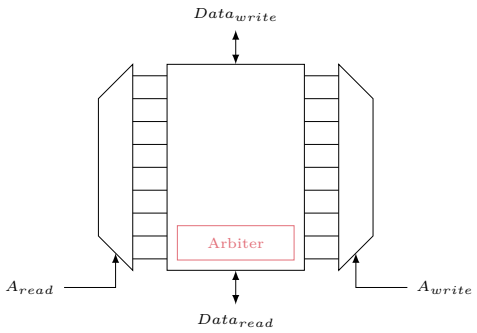
CSA - 9-bit Carray-Select-Addierer



Rechner/CSA

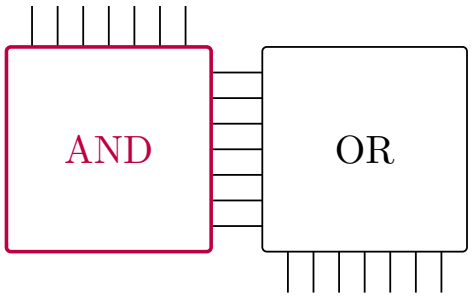


Rechner/DreiTorRegister

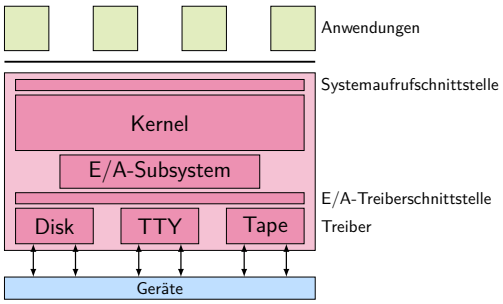


Rechner/Eintorspeicher

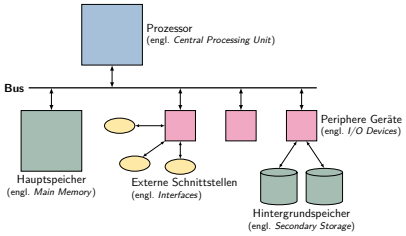
Rechner/GALPAL



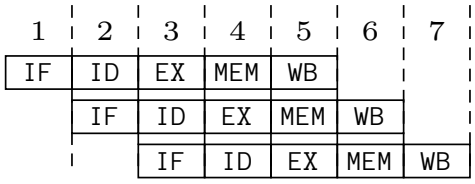
Rechner/Geraeteverwaltung



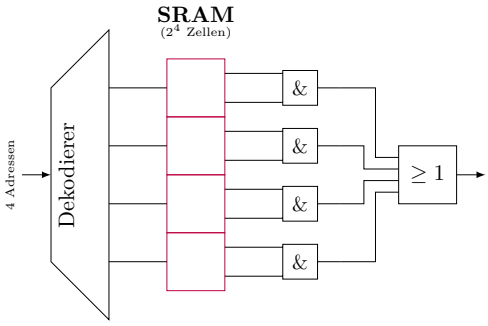
Rechner/HardwareSkizze



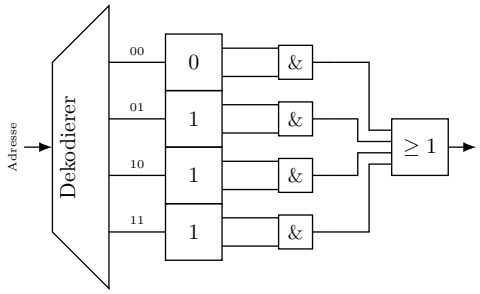
Rechner/ISA



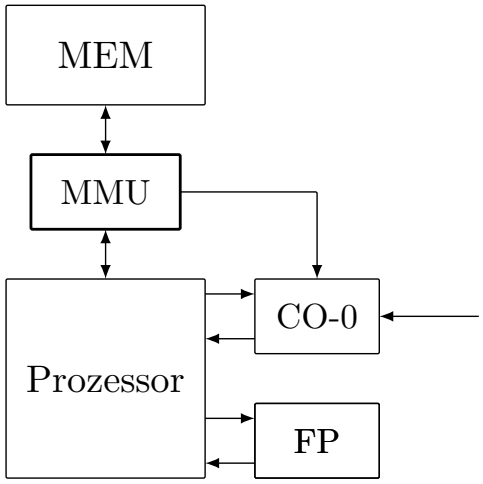
Rechner/LUT



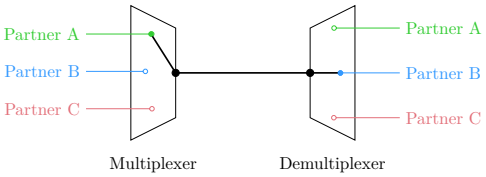
Rechner/LUTOder



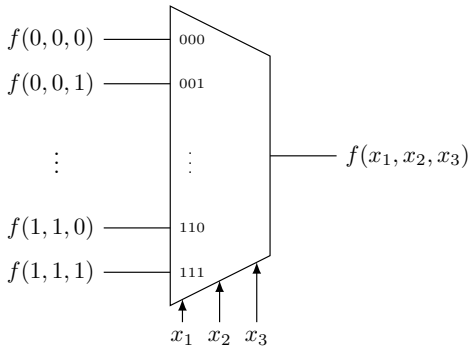
Rechner/MIPS



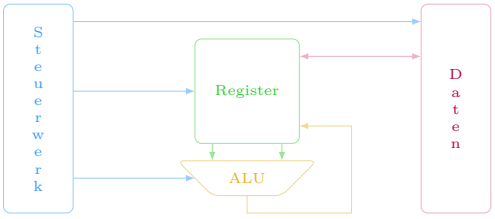
Rechner/MuxDemuxKommunikation

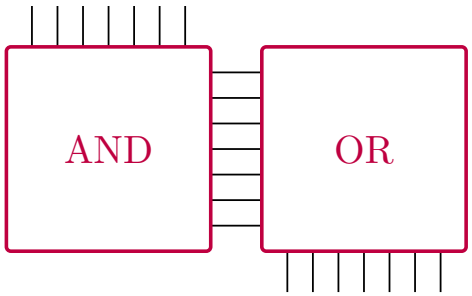


Rechner/MuxShannon

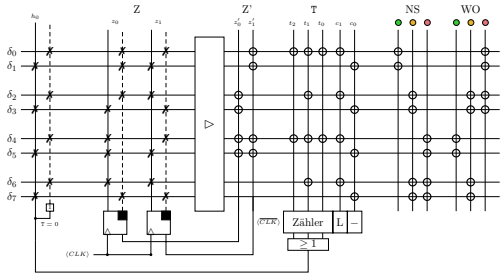


Rechner/NAdressmaschiene

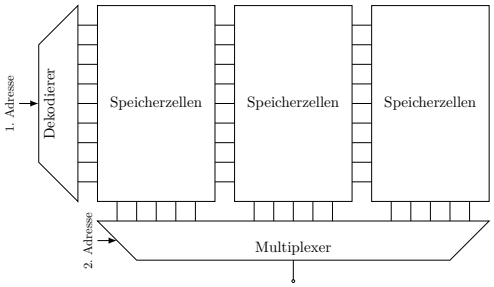




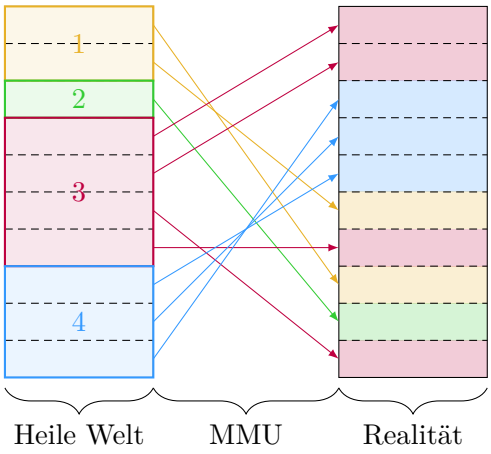
Rechner/PLA



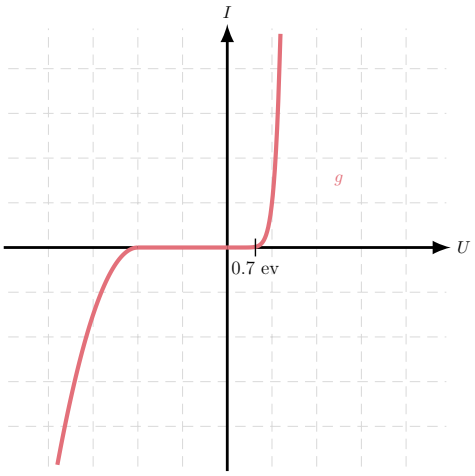
Rechner/PLAZuAmpel



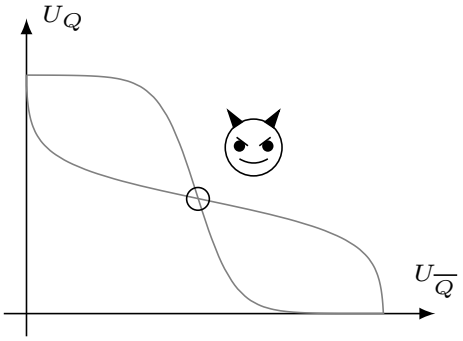
Rechner/PROM



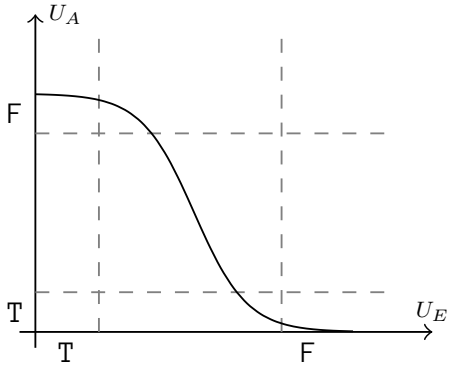
Rechner/Pages



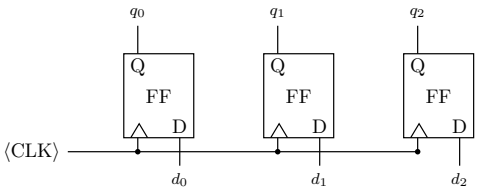
Rechner/Physik/DiodenStromstaerke



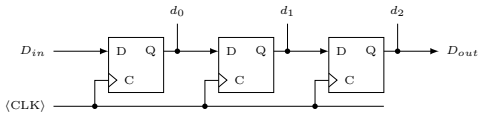
Rechner/Physik/Metastabil



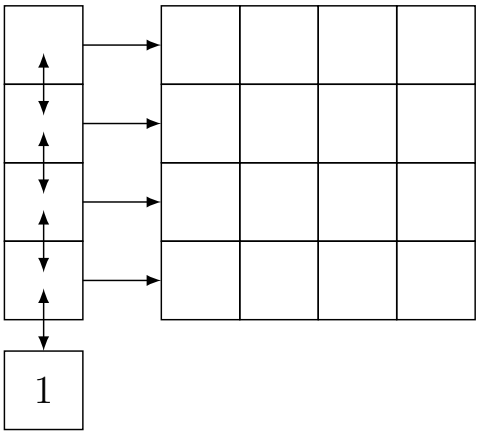
Rechner/Physik/TransistorStoertoleranz



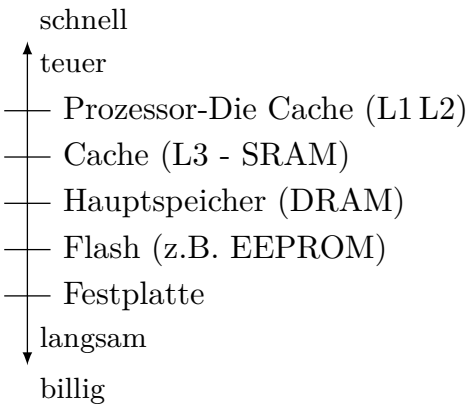
Rechner/RegisterParallel



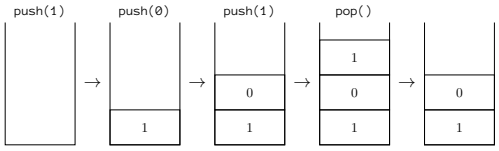
Rechner/RegisterSeriell



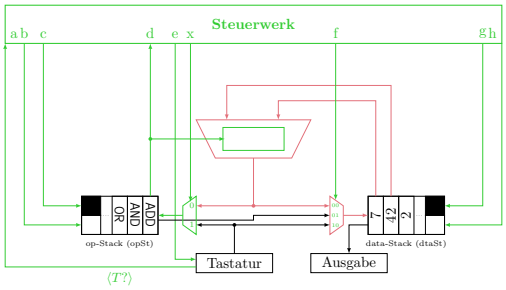
`Rechner/Shiftregister`



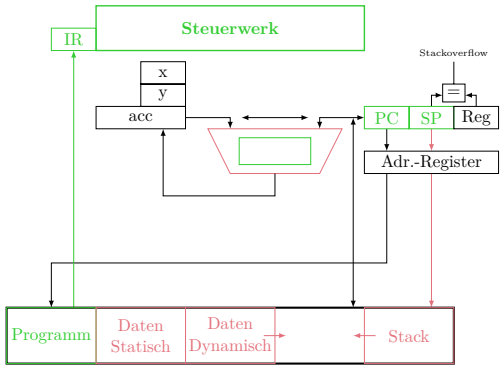
`Rechner/Speicherhierarchie`



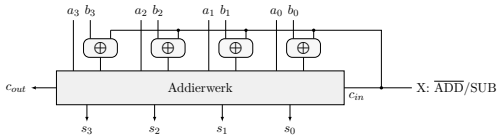
`Rechner/StackExample`



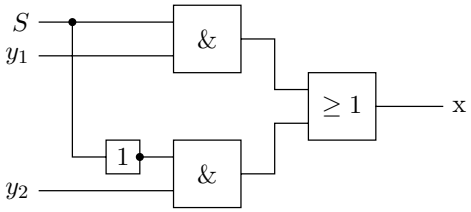
`Rechner/Stackmaschiene`



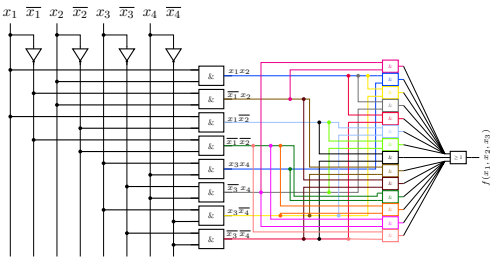
Rechner/StackmaschieneSimpler



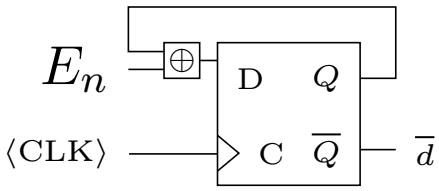
Schaltkreis/Addier-Subtrahierer



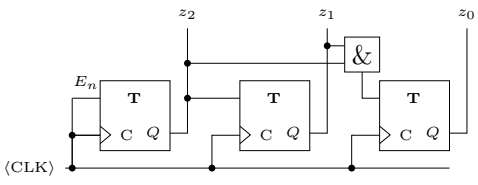
Schaltkreis/Demultiplexer



Schaltkreis/KomplexerSchaltkreis

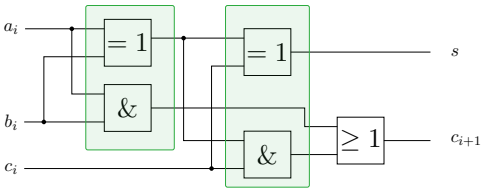


Schaltkreis/SynchronzaehlerDFF

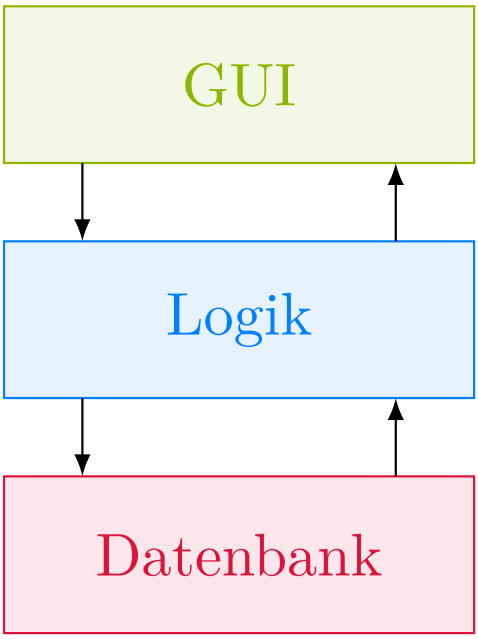


Schaltkreis/SynchronzaehlerTFF

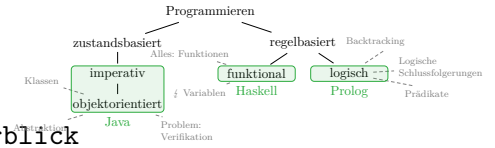
Schaltkreis/Volladdierer



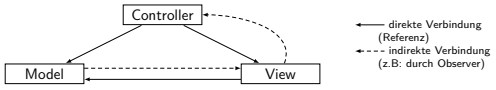
Software/DreiSchichtenArchitektur



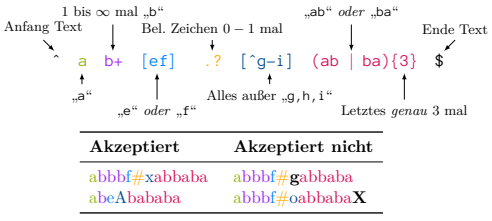
Software/Meta/ProgrammierparadigmenUeberblick



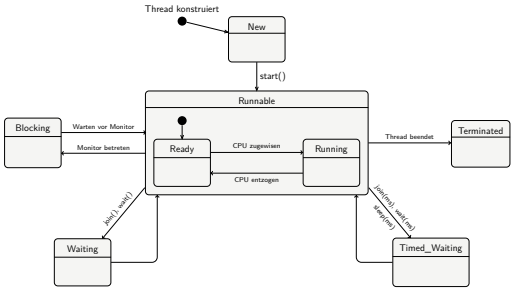
Software/ModelViewController

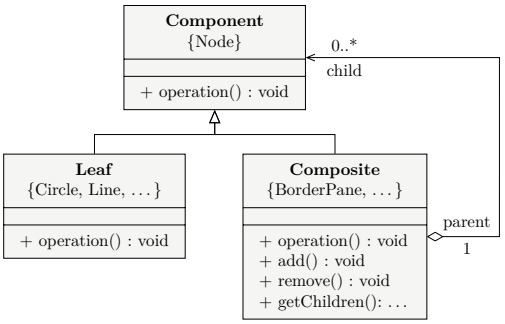


Software/RegexExample

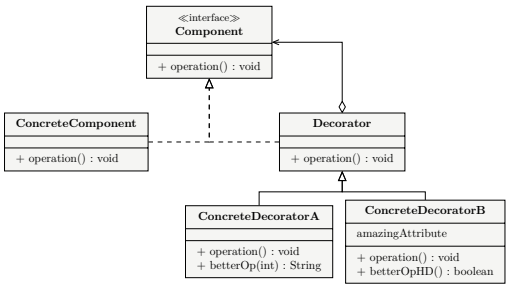


Software/ThreadStates

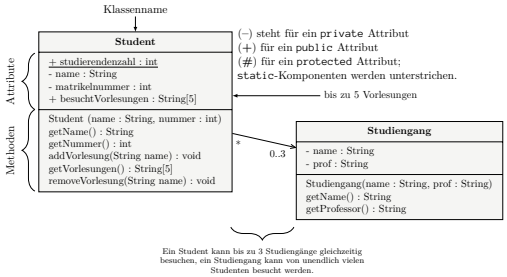




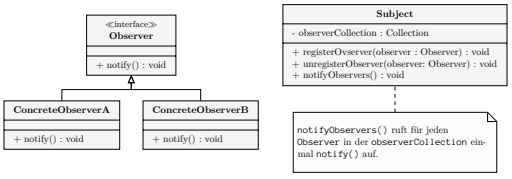
Software/UMLCompositePattern



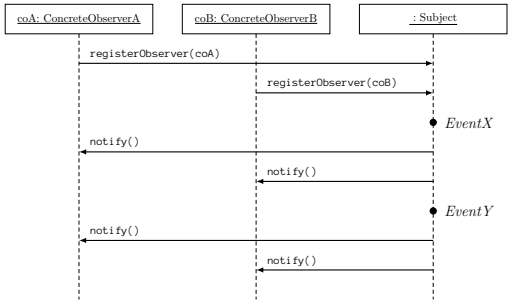
Software/UMLDecoratorPattern



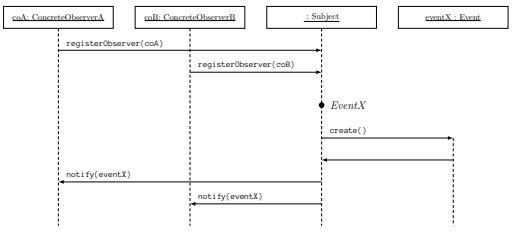
Software/UMLExample



Software/UMLObserverPattern

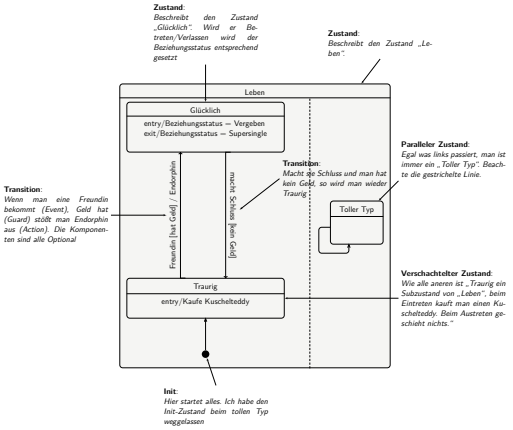


Software/UMLSEQObserverPattern

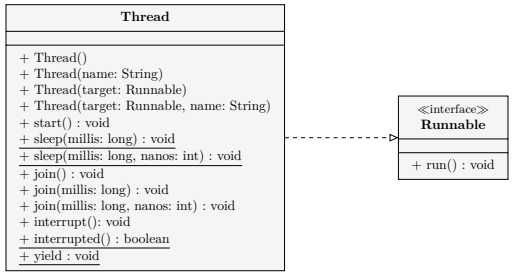


Software/UMLSEQObserverPatternAdapted

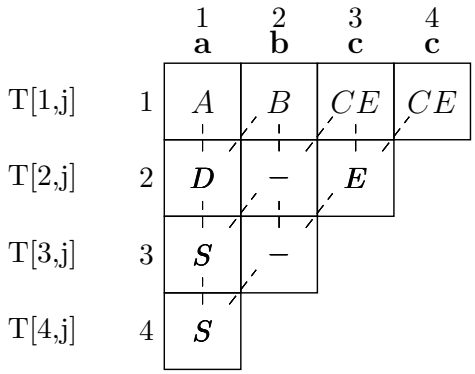
Software/UMLStateDiagramExample



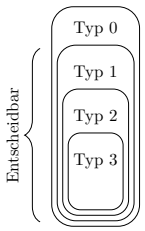
Software/UMLThread



Sprachen/CYKAlgorithmus



Sprachen/ChomskyHierarchie



Jede Grammatik - Aber **nicht** jede Sprache (Beispiel: π, alle irrationalen Zahlen)

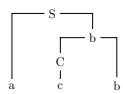
Kontextsensitiv:
$|w_1| \leq |w_2| \quad \forall w_1 \to w_2 \in P$

Kontextfrei:
$w_1 \in V \quad \forall w_1 \to w_2 \in P$

Regulär:
$w_2 \in \Sigma \cup \Sigma V$

Kann von NFA und DFA realisiert werden.

Sprachen/Grammatik



$G(\{S, B, C\}, \{a, b, c\}, P, S)$
P: $S \to aA$, $A \to Ck$, $C \to c$
Ableitung: acb