Logik für Informatiker

2. Aussagenlogik

Teil 7

20.05.2021

Viorica Sofronie-Stokkermans

Universität Koblenz-Landau

e-mail: sofronie@uni-koblenz.de

Anwendungsbereiche

1. Logikgatter
2. Puzzles
3. Planung
4. Verifikation

Anwendungsbereiche

1. Logikgatter

(Folien: “Schaltalgebra und Kombinatorische Logik ”

J. Kaiser, Univ. Magdeburg, Technische Informatik I, WS 2012/13)

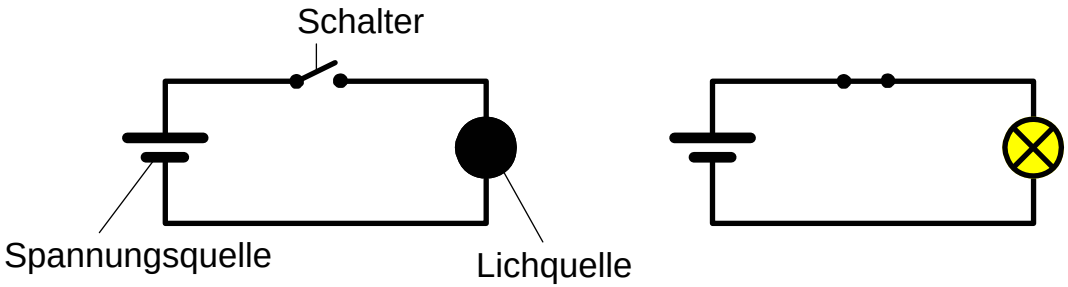
Schaltalgebra und kombinatorische Logik

1. Digitale elektrische Schaltungen
2. Beschreibung durch logische Ausdrücke
3. Boolesche Algebra
4. Schaltfunktionen
5. Synthese von Schaltungen
6. Schaltnetze

*Die Folien wurden unter Verwendung der Folien von A. Strey, Ulm, SS 03 erstellt

Digitale elektrische Schaltungen

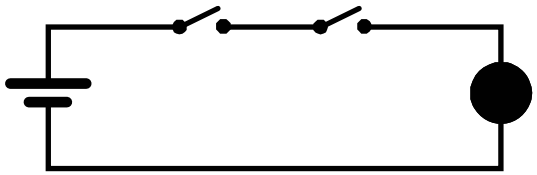
- eine einfache Schaltung:



- Strom fließt nur bei geschlossenem Stromkreis !
- zwei Zustände
  - Strom fließt $\Rightarrow$ Lampe leuchtet
  - Strom fließt nicht $\Rightarrow$ Lampe leuchtet nicht

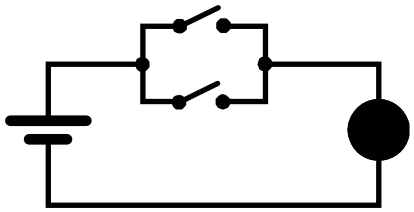
Digitale elektrische Schaltungen

- eine einfache Reihenschaltung:



⇒ Lampe leuchtet, wenn der erste und der zweite Schalter geschlossen werden

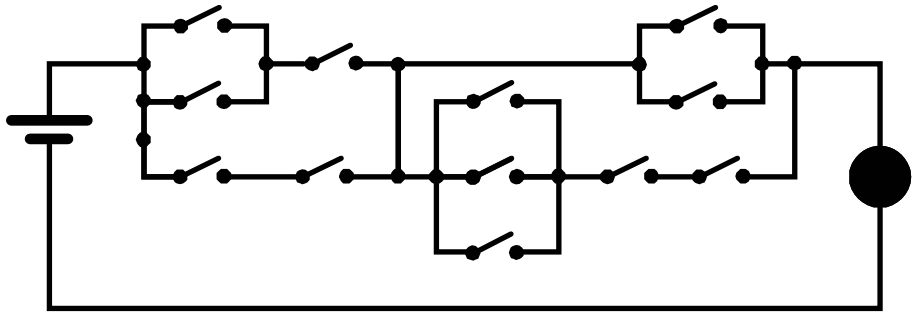
- eine einfache Parallelschaltung:



⇒ Lampe leuchtet, wenn der erste oder der zweite Schalter geschlossen wird

Digitale elektrische Schaltungen

- eine komplexe(re) Schaltung:



$\Rightarrow$ wann leuchtet die Lampe ?

$\Rightarrow$ wie kann man eine solche Schaltung beschreiben ?

Beschreibung durch logische Ausdrücke

- bei Betrachtung aus Sicht der Logik ergeben sich 2 Zustände:
  - Strom fließt / Strom fließt nicht
  - Lampe leuchtet / Lampe leuchtet nicht
  - an / aus
  - wahr / falsch
  - 1 / 0
  - 0 / 1
- im folgenden Betrachtung der Zustandsmenge {0,1}
- Zustände werden elektronisch realisiert:
  - Stromfluß / kein (oder ein sehr geringer) Stromfluß
  - Spannung / keine (oder eine sehr geringe) Spannung

Beschreibung durch logische Ausdrücke

Wie können logische Schaltungen beschrieben werden?
Welche logischen Grundverknüpfungen können identifiziert werden?

Logische Variablen nehmen den Wert „wahr“ oder „falsch“ an.
Andere Werte gibt es nicht: „Tertium non datur“

Verknüpfung von Variablen:

A	B	A UND B
falsch	falsch	falsch
falsch	wahr	falsch
wahr	falsch	falsch
wahr	wahr	wahr

Wahrheitstafel

Entwicklung:

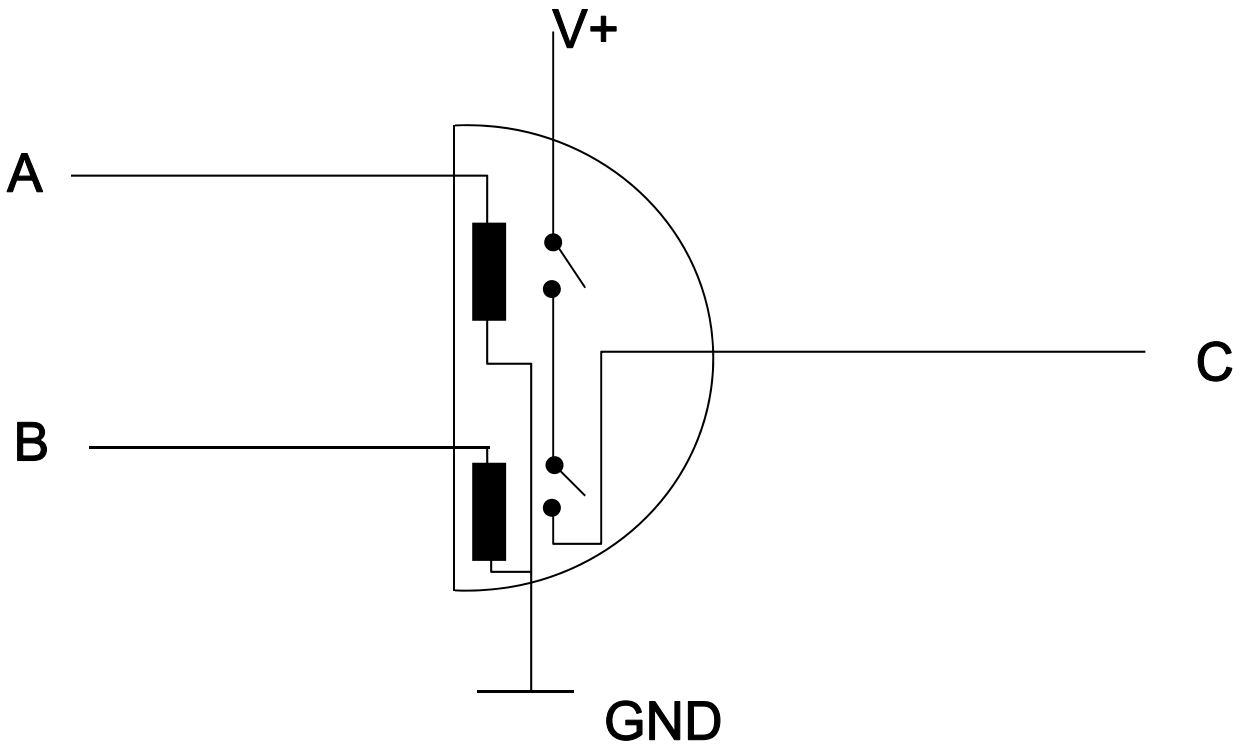
- Aristoteles 384-322 v.Chr. begründet „Syllogistik“ Lehre von den logischen Schlußformen
- Später bilden die Stoiker die Syllogistik als Aussagenlogik weiter aus.
- Im Mittelalter → Scholastik
- George Boole (1815-1864) → 1854 mathematische Formalisierung in „An Investigation of the Laws of Thought on which are founded the Mathematical Theories of Logic and Probabilities".
- Claude Shannon (1916-2001) hat im Rahmen seiner Diplomarbeit: „On the Symbolic Analysis of Relay and Switching Circuits (1940)“, gezeigt, dass man die Boolsche Algebra zur Beschreibung von Schaltkreisen anwenden kann.

Bild: http://de.wikipedia.org/wiki/Claude_Shannon

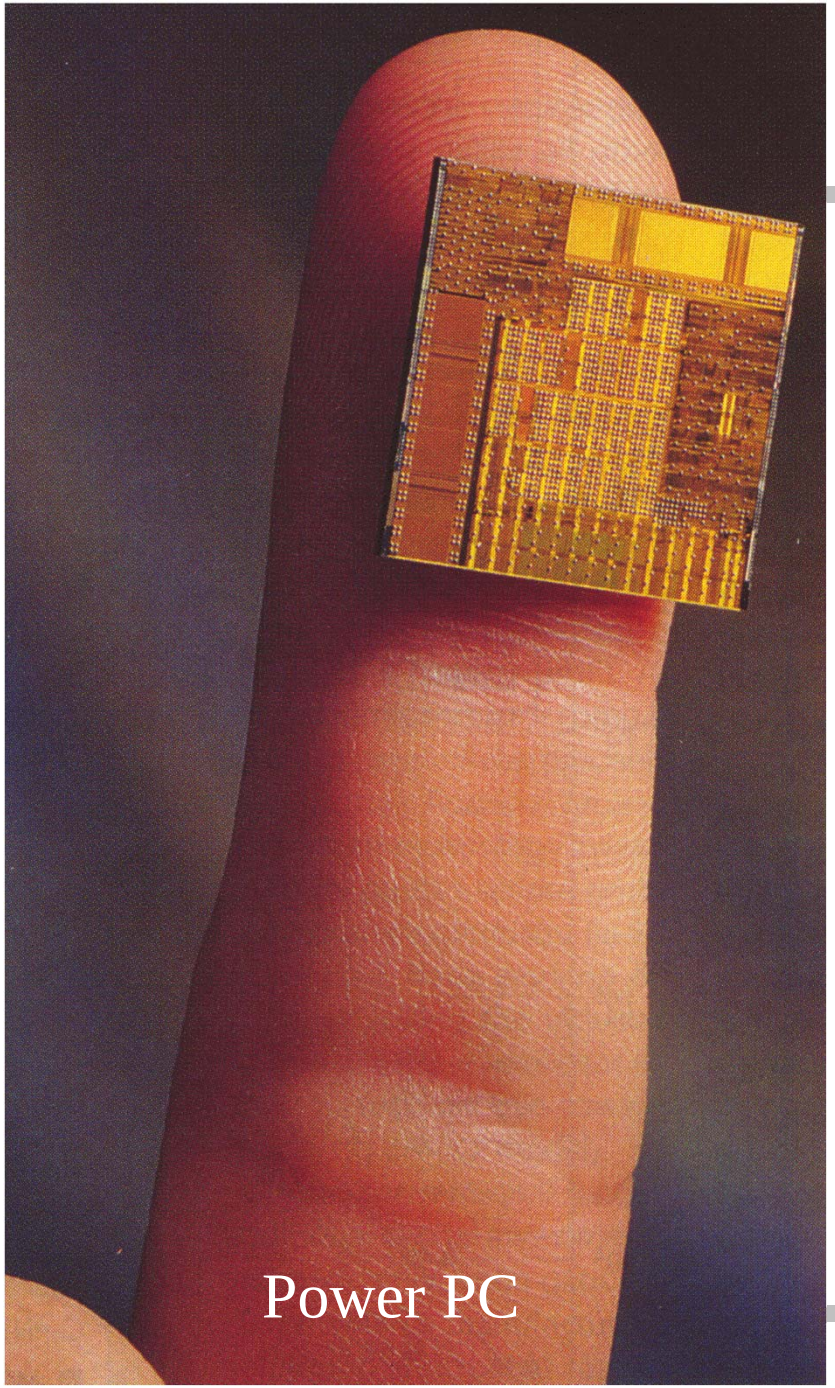
Wie realisiert man die Funktionen, die durch eine Wahrheitstafel beschrieben werden?

Realisierung einer UND-Funktion durch Gatter mit Relais:

A	B	A UND B
0	0	0
0	1	0
1	0	0
1	1	1



- elementare Funktionen auf {0,1} werden durch Gatter realisiert,
- komplexe Funktionen durch eine geeignete Verschaltung von mehreren Gattern

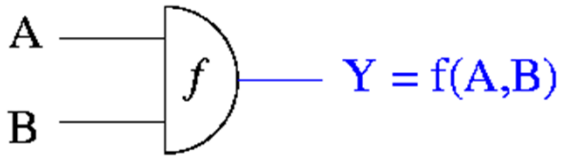


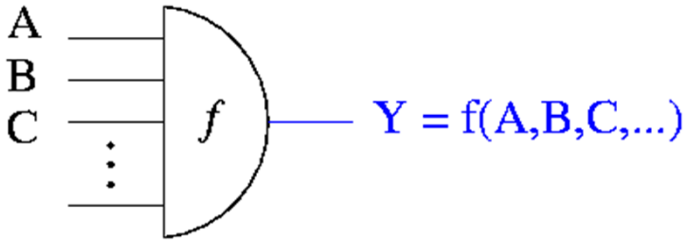
Realisierung durch
elektronische Schaltkreise:

klein
sparsam
schnell
zuverlässig
billig

Logische Schaltungen

- ein Gatter ist eine (elektrotechnische) „*Black Box*“ mit einem, zwei oder mehreren Eingängen A,B,C,... ∈ {0,1} und genau einem Ausgang Y∈{0,1} zur Realisierung einer Funktion Y = *f*(A,B,C,...)





A	B	Y
0	0	
0	1	
1	0	
1	1	

A	B	C	...	Y
0	0	0		
0	0	1		
⋮	⋮	⋮		
1	1	1		

- eine Wahrheitstabelle legt jeweils die Funktion fest.

Elementare Gatter

- ein UND-Gatter realisiert eine Konjunktion:
  - für zwei Eingänge:
    $Y = A \cdot B$
  - für *k* Eingänge:
    $Y = A_1 \cdot A_2 \cdot \ldots \cdot A_k$

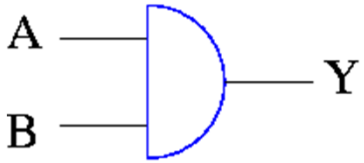
- Andere äquivalente Notationen:

  $Y = A \wedge B$

  $Y = AB$

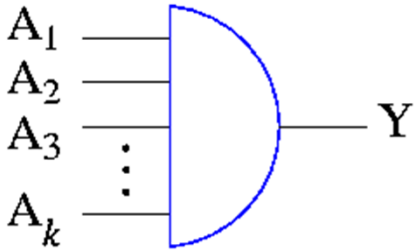
  $Y = A \,\&\, B$

für 2 Eingänge:

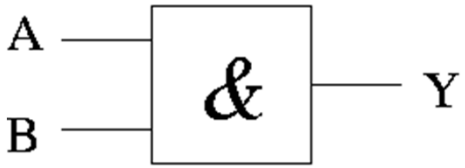


A	B	Y=A·B
0	0	0
0	1	0
1	0	0
1	1	1

für *k* Eingänge:



andere Darstellung:



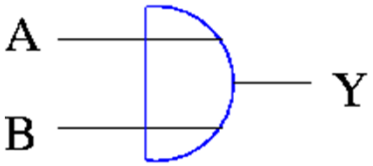
Elementare Gatter

- ein ODER-Gatter realisiert eine Disjunktion:
  - für zwei Eingänge: Y=A+B,
  - für *k* Eingänge: $Y=A_1+A_2+ \ldots +A_k$

- andere äquivalente Notationen:

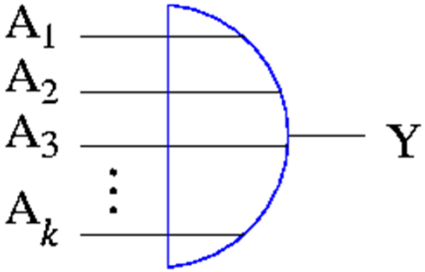
  $Y=A \vee B$

für 2 Eingänge:

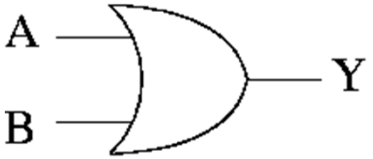


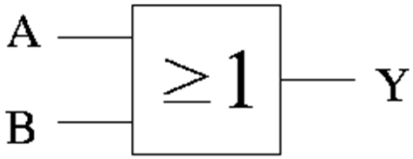
A	B	Y=A+B
0	0	0
0	1	1
1	0	1
1	1	1

für *k* Eingänge:



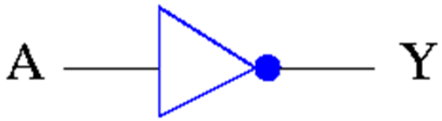
andere Darstellungen:





Elementare Gatter

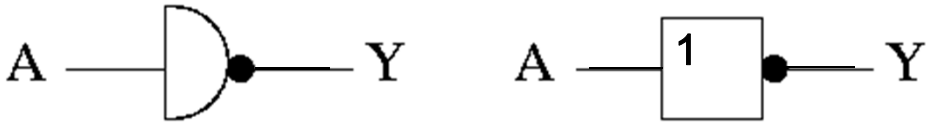
- ein NICHT-Gatter realisiert eine Negation:
  $Y=\overline{A}$



A	$Y=\overline{A}$
0	1
1	0

- andere äquivalente Notationen:
  Y=A'
  Y= ¬A

andere Darstellungen:



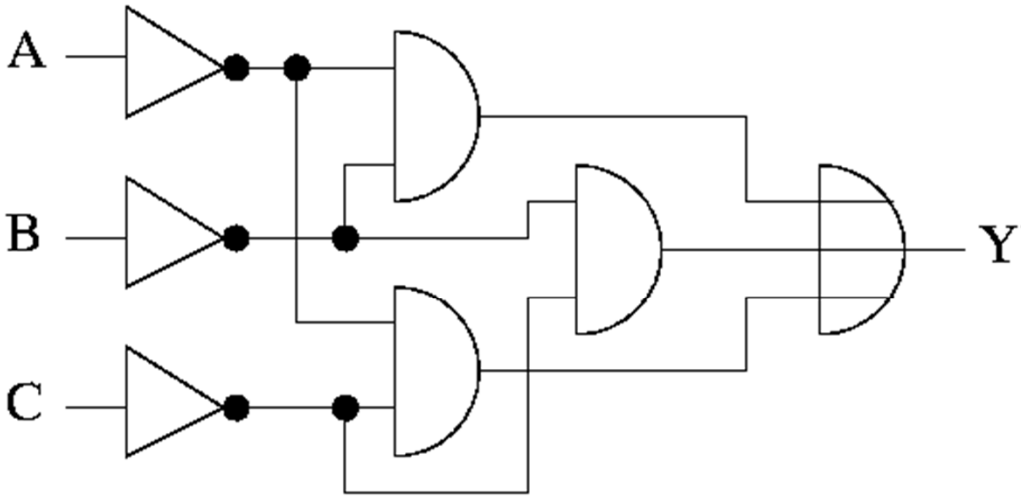
Beispiel einer logischer Schaltung

- Gesucht ist eine Schaltung, die eine logische 1 generiert, wenn höchstens einer von drei Eingängen A,B,C den Wert 1 aufweist.
- erste Lösungsvariante:

Wahrheitstabelle:

A	B	C	Y
0	0	0	1
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	0

Realisierung:



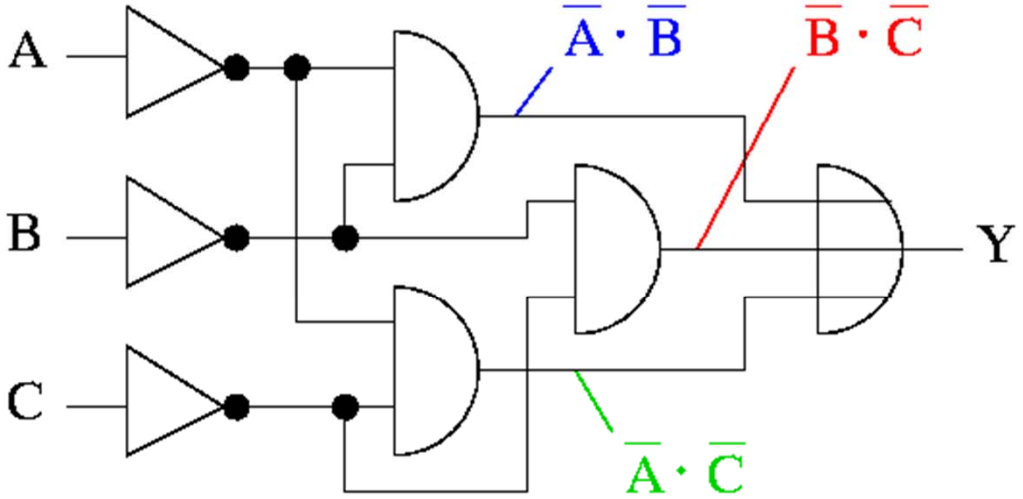
Beispiel einer logischen Schaltung

- Gesucht ist eine Schaltung, die eine logische 1 generiert, wenn höchstens einer von drei Eingängen A,B,C den Wert 1 aufweist.
- erste Lösungsvariante:

Wahrheitstabelle:

A	B	C	Y
0	0	0	1
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	0

Realisierung:



$$Y = \overline{A} \cdot \overline{B} + \overline{B} \cdot \overline{C} + \overline{A} \cdot \overline{C}$$

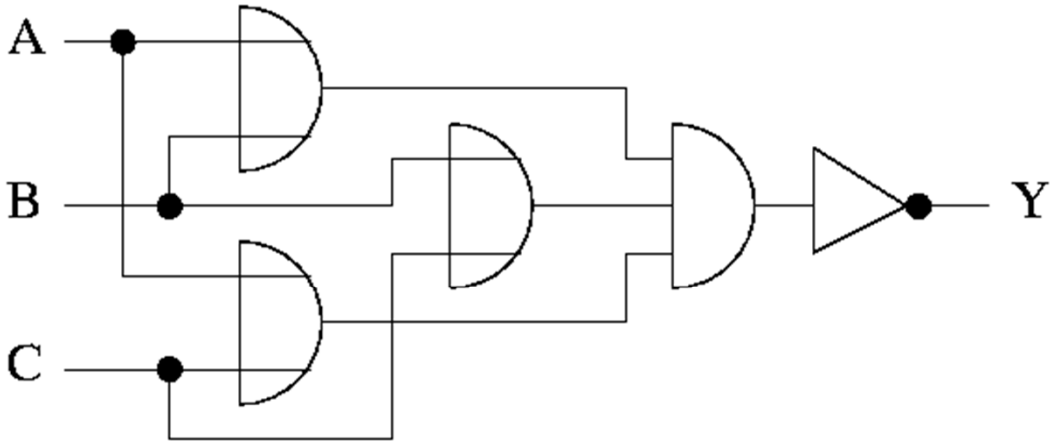
Beispiel einer logischer Schaltung

- zweite Lösungsvariante:

Wahrheitstabelle:

A	B	C	Y
0	0	0	1
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	0

Realisierung:



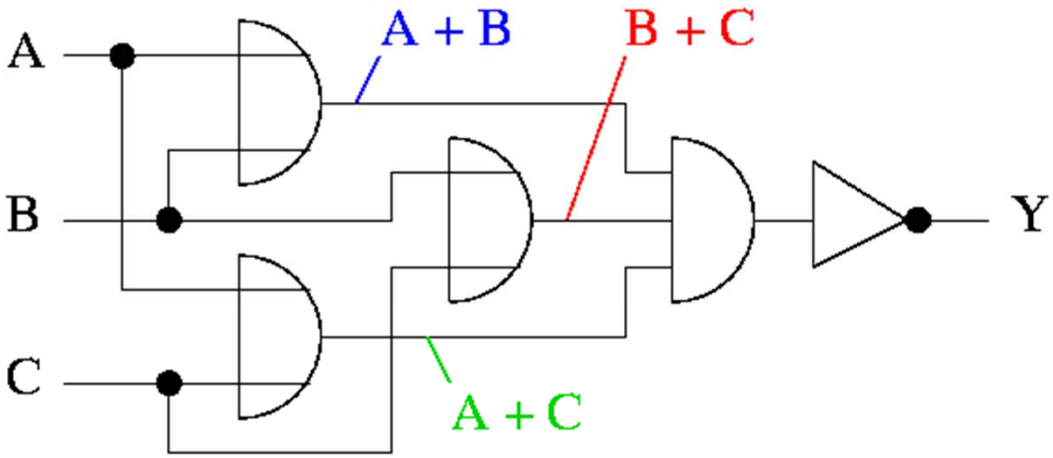
Beispiel einer logischer Schaltung

- zweite Lösungsvariante:

Wahrheitstabelle:

A	B	C	Y
0	0	0	1
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	0

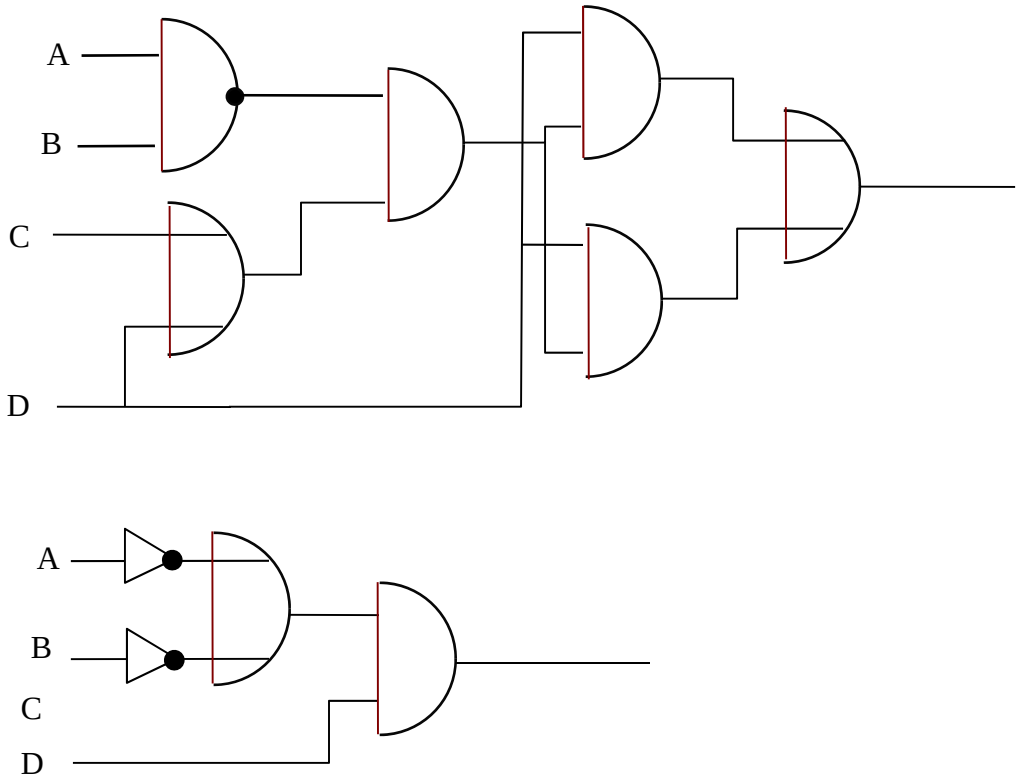
Realisierung:



$$Y = \overline{(A + B) \cdot (B + C) \cdot (A + C)}$$

- welche Variante ist besser ?

Äquivalenz von Schaltnetzen



??

Schaltfunktionen

- Funktionen $f: \{0,1\}^n \rightarrow \{0,1\}^m$ mit $n,m \geq 1$ werden auch als Schaltfunktionen bezeichnet
- Eine Schaltfunktion $f: \{0,1\}^n \rightarrow \{0,1\}$ heißt eine n-stellige Boolesche Funktion
- Jede Schaltfunktion $f: \{0,1\}^n \rightarrow \{0,1\}^m$ kann durch m Boolesche Funktionen ausgedrückt werden
- Jede Boolesche Funktion läßt sich eindeutig beschreiben
  - durch eine Wahrheitstabelle (auch Wahrheitstafel genannt)
  - durch einen booleschen Ausdruck (gebildet durch Boolesche Variablen und Operationen aus der Booleschen Algebra)
- Es gibt 2^{2^n} verschiedene n-stellige Boolesche Funktionen (also 16 zweistellige, 256 dreistellige, 65536 vierstellige, ...)

$|A| = n \qquad |B| = m$

$$\left| \{ f \mid f : A \to B \} \right| = m^n$$

Beweis: $A = \{a_1 \dots a_n\} \qquad B = \{b_1 \dots b_m\}$

x	a_1	a_2	$\dots$	a_n
$f(x)$	↑	↑	$\dots$	↑

m Möglichkeiten $\quad \dots \quad$ m Möglichkeiten

$f(a_1) \in \{b_1 \dots b_m\} \quad \dots \quad f(a_n) \in \{b_1 \dots b_m\}$.

$\Rightarrow \underbrace{m \cdot m \cdot \dots \cdot m}_{n \text{ Mal}}$ Funktionen $f : A \to B$.

Synthese

- eine KDNF ist günstiger als eine KKNF, wenn nur für wenige Kombinationen der Eingabewerte $f(x_1, x_2, ..., x_n) = 1$ gilt.
  Umgekehrt, wenn nur für wenige Kombinationen der Eingabewerte $f(x_1, x_2, ..., x_n) = 0$ gilt, ist die KKNF vorzuziehen.

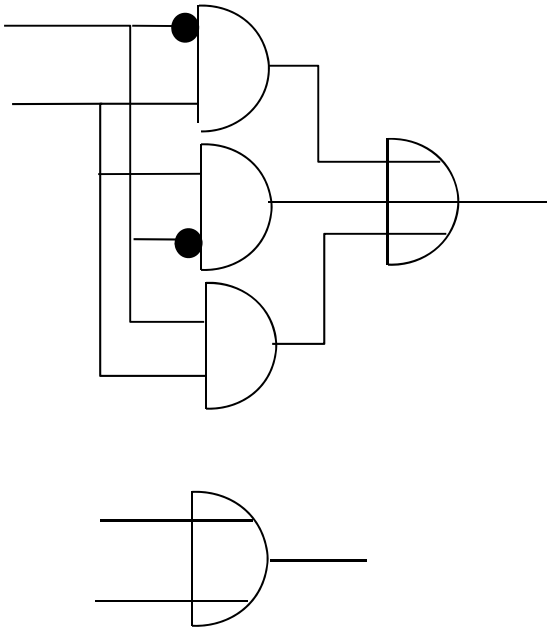
- Beispiel : ODER-Funktion

Wahrheitstafel

x	*y*	*f(x,y)*
0	0	0
0	1	1
1	0	1
1	1	1

KDNF: $\bar{x}\cdot y + x\cdot\bar{y} + x\cdot y$

KKNF: $x + y$



Anwendungsbereiche

2. Puzzles

2.1. Logische Puzzles

Wahrheit und Lügen

2.1. Logische Puzzles

Hier ist ein berühmtes Rätsel:
Lediglich eine dieser drei Personen sagt die Wahrheit, die anderen beiden lügen. Kannst du anhand ihrer Aussagen herausfinden, wer die Wahrheit sagt?

A: Ich lüge nie.
B: A lügt. Ich sage hier die Wahrheit!
C: B lügt. Ich bin der Ehrliche von uns!

2.1. Logische Puzzles

Hier ist ein berühmtes Rätsel:
Lediglich eine dieser drei Personen sagt die Wahrheit,
die anderen beiden lügen. Kannst du anhand ihrer
Aussagen herausfinden, wer die Wahrheit sagt?

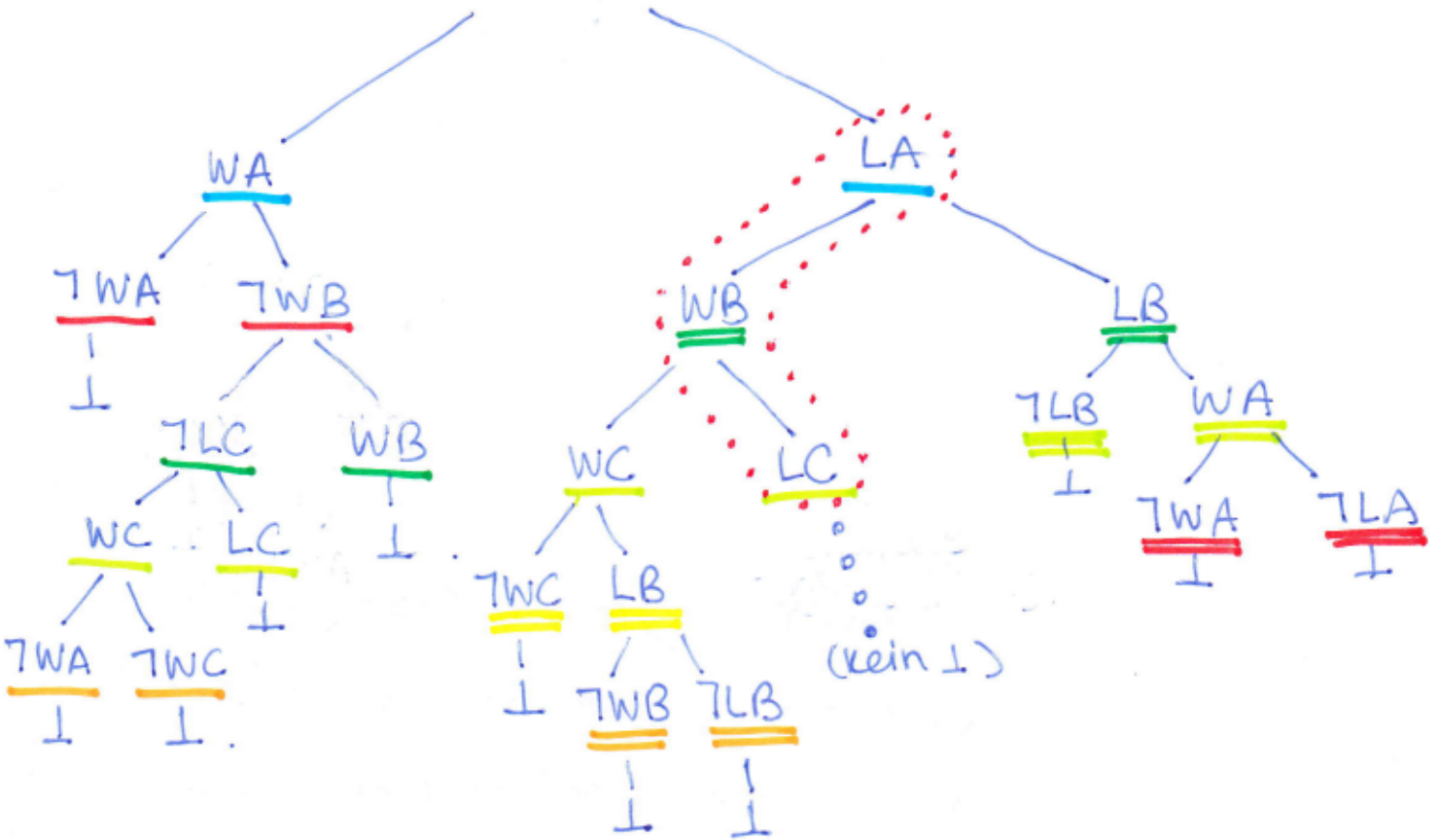
A: Ich lüge nie.
B: A lügt. Ich sage hier die Wahrheit!
C: B lügt. Ich bin der Ehrliche von uns!

$(WA \vee WB \vee WC) \wedge \neg(WA \wedge WB) \wedge \neg(WA \wedge WC) \wedge \neg(WB \wedge WC) \wedge$
$(WA \vee LA) \wedge (WB \vee LB) \wedge (WC \vee LC) \wedge \neg(WA \wedge LA) \wedge \neg(WB \wedge LB) \wedge \neg(WC \wedge LC) \wedge$

$(WA \rightarrow LB) \wedge (WA \rightarrow LC)$
$(WB \rightarrow LA) \wedge (LB \rightarrow WA) \wedge (WB \rightarrow LC) \wedge$
$(WC \rightarrow LB) \wedge (WC \rightarrow LA) \wedge (LC \rightarrow WB)$

Ist die Formel erfüllbar? Modell?

$$(WA \vee WB \vee WC) \wedge \neg(WA \wedge WB) \wedge \neg(WA \wedge WC) \wedge \neg(WB \wedge WC)$$
$$\wedge (WA \vee LA) \wedge (WB \vee LB) \wedge (WC \vee LC)$$
$$\wedge \neg(WA \wedge LA) \wedge \neg(WB \wedge LB) \wedge \neg(WC \wedge LC)$$
$$\wedge (WB \rightarrow LA) \wedge (LB \rightarrow WA)$$
$$\wedge (WC \rightarrow LB) \wedge (LC \rightarrow WB)$$



2.1 Logische Puzzles

“Gibt es Superman?”

aus Christoph Drösser, “Der Logik-Verführer”

2.1 Logische Puzzles

- E steht für: "Superman existiert"
- L steht für: "Superman ist in der Lage, Böses zu verhindern"
- W steht für: "Superman ist willig, Böses zu verhindern"
- U steht für: "Superman ist unfähig"
- B steht für: "Superman ist bösartig"
- V steht für: "Superman verhindert Böses"

"Wenn Superman in der Lage und willig ist, etwas Böses zu verhindern, dann verhindert er es".

(1) $(L \wedge W) \rightarrow V$

"Wenn Superman nicht in der Lage ist, etwas Böses zu verhindern, dann ist er unfähig".

(2) $\neg L \rightarrow U$

2.1 Logische Puzzles

- E steht für: "Superman existiert"
- L steht für: "Superman ist in der Lage, Böses zu verhindern"
- W steht für: "Superman ist willig, Böses zu verhindern"
- U steht für: "Superman ist unfähig"
- B steht für: "Superman ist bösartig"
- V steht für: "Superman verhindert Böses"

"Wenn Superman nicht das Böse verhindern will, dann ist er bösartig".

(3) $\neg W \rightarrow B$

"Superman verhindert nicht das Böse"

(4) $\neg V$

"Wenn Superman existiert, dann ist er weder bösartig noch unfähig"

(5) $E \rightarrow (\neg B \wedge \neg U)$

2.1 Logische Puzzles

- E steht für: "Superman existiert"
- L steht für: "Superman ist in der Lage, Böses zu verhindern"
- W steht für: "Superman ist willig, Böses zu verhindern"
- U steht für: "Superman ist unfähig"
- B steht für: "Superman ist bösartig"
- V steht für: "Superman verhindert Böses"

Ist die folgende Formel erfüllbar?

$$
\begin{array}{ll}
((L \wedge W) \rightarrow V) & \wedge \\
(\neg L \rightarrow U) & \wedge \\
(\neg W \rightarrow B) & \wedge \\
\neg V & \wedge \\
(E \rightarrow (\neg B \wedge \neg U)) & \wedge \\
E &
\end{array}
$$

2.1 Logische Puzzles

Ist die folgende Formel erfüllbar?

$$
\begin{array}{ll}
((L \wedge W) \rightarrow V) & \wedge \\
(\neg L \rightarrow U) & \wedge \\
(\neg W \rightarrow B) & \wedge \\
\neg V & \wedge \\
(E \rightarrow (\neg B \wedge \neg U)) & \wedge \\
E &
\end{array}
$$

KNF (Mengenschreibweise)

(1) $\{\neg L, \neg W, V\}$
(2) $\{L, U\}$
(3) $\{W, B\}$
(4) $\{\neg V\}$
(5) $\{\neg E, \neg B\}$
(6) $\{\neg E, \neg U\}$
(7) $\{E\}$

2.1 Logische Puzzles

(1)	$\{\neg L, \neg W, V\}$	gegeben
(2)	$\{L, U\}$	gegeben
(3)	$\{W, B\}$	gegeben
(4)	$\{\neg V\}$	gegeben
(5)	$\{\neg E, \neg B\}$	gegeben
(6)	$\{\neg E, \neg U\}$	gegeben
(7)	$\{E\}$	gegeben
(8)	$\{\neg U\}$	aus (6), (7)
(9)	$\{L\}$	*aus*(2), (8)
(10)	$\{\neg W, V\}$	aus (1), (9)
(11)	$\{\neg W\}$	aus (4), (10)
(12)	$\{B\}$	aus (3), (11)
(13)	$\{\neg E\}$	aus (5), (12)
(14)	$\bot$	aus (7), (13)

2.2. Sudoku

	3							
			1	9	5			
	9	8					6	
8				6				
4					3			1
				2				
	6					2	8	
			4	1	9			5
							7	

2.2. Sudoku

	3							
			1	9	5			
	9	8					6	
8				6				
4					3			1
				2				
	6					2	8	
			4	1	9			5
							7	

5	3	4	6	7	8	9	1	2
6	7	2	1	9	5	3	4	8
1	9	8	3	4	2	5	6	7
8	5	9	7	6	1	4	2	3
4	2	6	8	5	3	7	9	1
7	1	3	9	2	4	8	5	6
9	6	1	5	3	7	2	8	4
2	8	7	4	1	9	6	3	5
3	4	5	2	8	6	1	7	9

Aussagenlogisches Modell

Koordinaten der Felder:

$Feld(i, j)$ ist das Feld in Zeile i und Spalte j.

Aussagenvariablen:

$P_{i,j,k}$ für $i, j, k \in \{1, 2, \ldots, 9\}$.

Steht für "Feld mit den Koordinaten (i , j) enthält eine k".

Wertebelegungen beschreiben Beschriftungen des Gitters.

Ziel:

Für jede Anfangsbeschriftung eine Formelmenge F, so dass für alle Wertebelegungen $\mathcal{A}$ gilt:
$\mathcal{A} \models F$ genau dann, wenn $\mathcal{A}$ beschreibt eine korrekte Lösung.

Aussagenlogisches Modell

Wir beschreiben zunächst eine Formelmenge F_1, die die Grundregeln des Spiels beschreibt.

Beschriftungen:

"Auf jedem Feld steht mindestens eine Zahl."

$$\phi_1 := \bigwedge_{i,j=1}^{9} \bigvee_{k=1}^{9} P_{i,j,k}$$

"Auf jedem Feld steht höchstens eine Zahl."

$$\phi_2 := \bigwedge_{i,j=1}^{9} \bigwedge_{\substack{k,l=1 \\ k \neq l}}^{9} \neg(P_{i,j,k} \wedge P_{i,j,l})$$

Aussagenlogisches Modell

Zeilen

"Jede Zahl kommt in jeder Zeile vor"

$\phi_3 := \bigwedge_{i,k=1}^{9} \bigvee_{j=1}^{9} P_{i,j,k}$

Spalten

"Jede Zahl kommt in jeder Spalte vor"

$\phi_4 := \bigwedge_{j,k=1}^{9} \bigvee_{i=1}^{9} P_{i,j,k}$

Blöcke

"Jede Zahl kommt in jedem Block vor"

$$\phi_5 := \bigwedge_{i,j=0}^{2} \bigwedge_{k=1}^{9} \bigvee_{i',j'=1}^{9} P_{3*i+i',3j+j',k}$$

Aussagenlogisches Modell

Zeilen

"Jede Zahl kommt in jeder Zeile vor"

$\phi_3 := \bigwedge_{i,k=1}^{9} \bigvee_{j=1}^{9} P_{i,j,k}$

Spalten

"Jede Zahl kommt in jeder Spalte vor"

$\phi_4 := \bigwedge_{j,k=1}^{9} \bigvee_{i=1}^{9} P_{i,j,k}$

Blöcke

"Jede Zahl kommt in jedem Block vor"

$$\phi_5 := \bigwedge_{i,j=0}^{2} \bigwedge_{k=1}^{9} \bigvee_{i',j'=1}^{9} P_{3*i+i',3j+j',k}$$

Anfangsbeschriftung

$F = F_1 \cup \{P_{i,j,k} \mid Feld(i,j)$ ist mit k beschriftet$\}$

Aussagenlogisches Modell

Zeilen

"Jede Zahl kommt in jeder Zeile vor"

$\phi_3 := \bigwedge_{i,k=1}^{9} \bigvee_{j=1}^{9} P_{i,j,k}$

Spalten

"Jede Zahl kommt in jeder Spalte vor"

$\phi_4 := \bigwedge_{j,k=1}^{9} \bigvee_{i=1}^{9} P_{i,j,k}$

Blöcke

"Jede Zahl kommt in jedem Block vor"

$$\phi_5 := \bigwedge_{i,j=0}^{2} \bigwedge_{k=1}^{9} \bigvee_{i',j'=1}^{9} P_{3*i+i',3j+j',k}$$

Anfangsbeschriftung

$F = F_1 \cup \{P_{i,j,k} \mid Feld(i,j) \text{ ist mit } k \text{ beschriftet}\}$

Modell von F: $\mapsto$ korrekte Lösung.

Frage: Ist F erfüllbar?

Überprüfung: Resolution oder Tableaux

Anwendungsbereiche

3. Planen

Ähnliches Problem: Planen

Beispiel: Modellierung eines Zeitplanunge-Problems

An einer Schule gibt es drei Lehrer mit folgenden Fächerkombinationen:

Müller:	Mathe
Schmidt:	Deutsch
Körner:	Mathe, Deutsch

Es soll folgender Lehrplan erfüllt werden:

	Klasse a)	Klasse b)
Stunde I	Mathe	Deutsch
Stunde II	Deutsch	Deutsch
Stunde III	Mathe	Mathe

Dabei soll jeder Lehrer mindestens 2 Stunden unterrichten

Ähnliches Problem: Planen

Beispiel: Modellierung eines Zeitplanunge-Problems

An einer Schule gibt es drei Lehrer mit folgenden Fächerkombinationen:

Müller:	Mathe
Schmidt:	Deutsch
Körner:	Mathe, Deutsch

Es soll folgender Lehrplan erfüllt werden:

	Klasse a)	Klasse b)
Stunde I	Mathe	Deutsch
Stunde II	Deutsch	Deutsch
Stunde III	Mathe	Mathe

Modellierung: Dabei soll jeder Lehrer mindestens 2 Stunden unterrichten

Aussagenvariablen: $P_{s,k,N,f}$ "Lehrer N unterrichtet Fach f f. Klasse k in Stunde s"

Regeln: $(P_{1,a,M,m} \vee P_{1,a,K,m}) \wedge (P_{1,b,S,d} \vee P_{1,b,K,d})$

$(P_{2,a,S,d} \vee P_{2,a,K,d}) \wedge (P_{2,b,S,d} \vee P_{2,b,K,d})$

$(P_{3,a,M,m} \vee P_{3,a,K,m}) \wedge (P_{3,b,S,d} \vee P_{3,a,K,d})$

$\neg(P_{1,a,K,m} \wedge P_{1,b,K,d}) \wedge \neg(P_{2,a,K,d} \wedge P_{2,b,K,d}) \wedge \neg(P_{2,a,S,d} \wedge P_{2,b,S,d}) \wedge$

$\neg(P_{3,a,K,m} \wedge P_{3,b,K,m}) \wedge (P_{1,a,M,m} \wedge P_{1,b,M,m}) \ldots$

Anwendungsbereiche

4. Verifikation

(Folien aus "Angewandte Logik: Von Tante Agatha zu korrekter Software"
Viorica Sofronie-Stokkermans und Uwe Waldmann
Day of the open doors, 2004, MPI für Informatik)

Beispiel 3: Handy

Ein Handy kann sich in verschiedenen Zuständen befinden.
Beispielsweise:

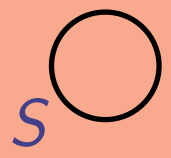
- Startzustand (direkt nach dem Einschalten).
- Benutzer gibt Telefonnummer ein.
- Telefonnummer ist vollständig eingegeben;
  Handy versucht, Basisstation zu erreichen.
- Kontakt zur Basisstation aufgenommen;
  anderer Teilnehmer wird angerufen.
- Gespräch wird geführt.

  ...

Beispiel 3: Handy

Wir stellen die Zustände durch Kreise dar.

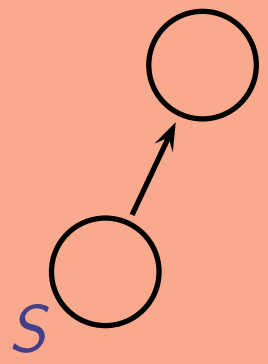
Wenn es möglich ist, von einem Zustand in einen anderen zu wechseln, zeichnen wir einen Pfeil dazwischen:



Beispiel 3: Handy

Wir stellen die Zustände durch Kreise dar.

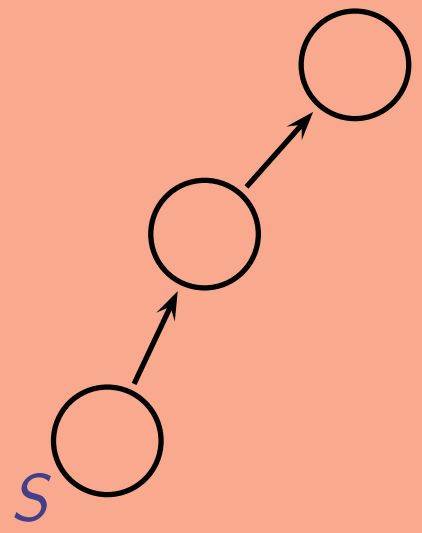
Wenn es möglich ist, von einem Zustand in einen anderen zu wechseln, zeichnen wir einen Pfeil dazwischen:



Beispiel 3: Handy

Wir stellen die Zustände durch Kreise dar.

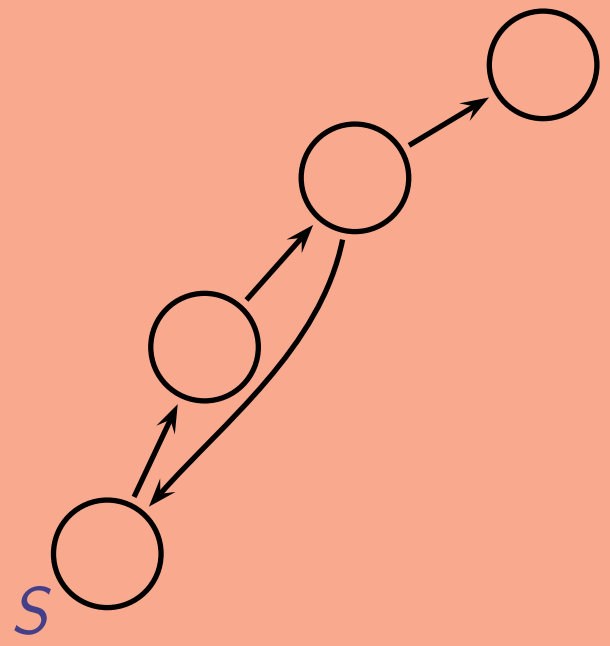
Wenn es möglich ist, von einem Zustand in einen anderen zu wechseln, zeichnen wir einen Pfeil dazwischen:



Beispiel 3: Handy

Wir stellen die Zustände durch Kreise dar.

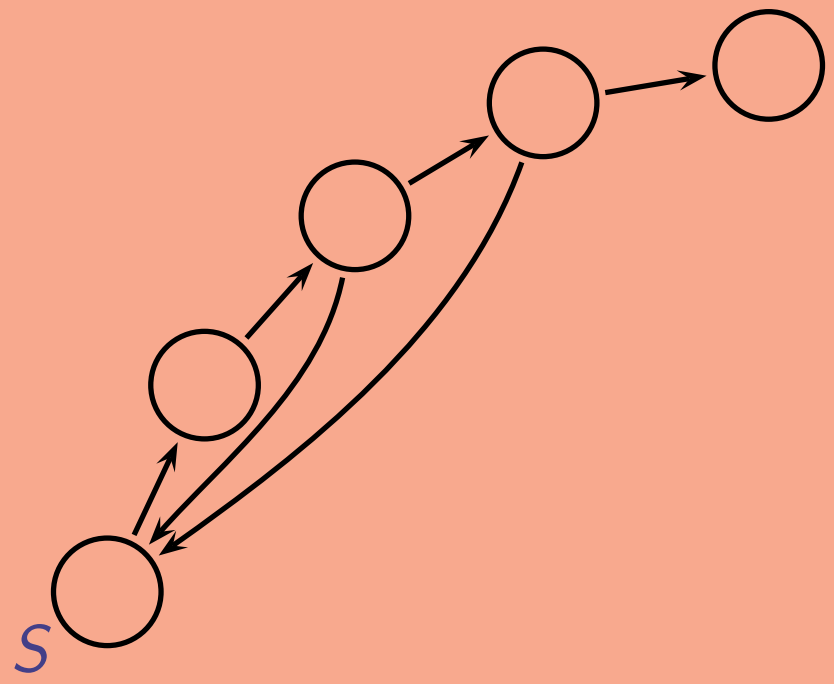
Wenn es möglich ist, von einem Zustand in einen anderen zu wechseln, zeichnen wir einen Pfeil dazwischen:



Beispiel 3: Handy

Wir stellen die Zustände durch Kreise dar.

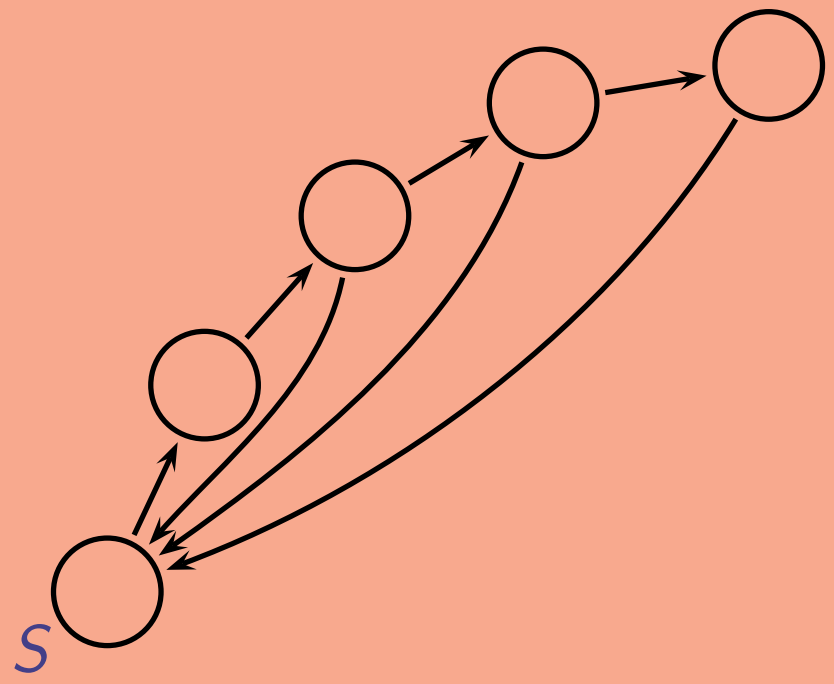
Wenn es möglich ist, von einem Zustand in einen anderen zu wechseln, zeichnen wir einen Pfeil dazwischen:



Beispiel 3: Handy

Wir stellen die Zustände durch Kreise dar.

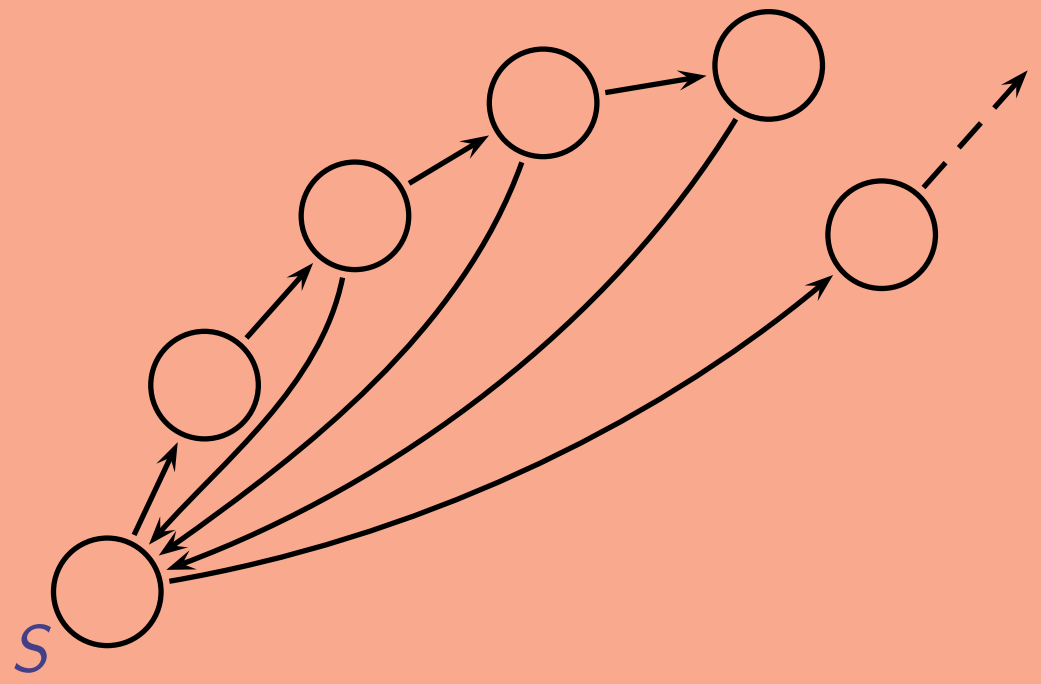
Wenn es möglich ist, von einem Zustand in einen anderen zu wechseln, zeichnen wir einen Pfeil dazwischen:



Beispiel 3: Handy

Wir stellen die Zustände durch Kreise dar.

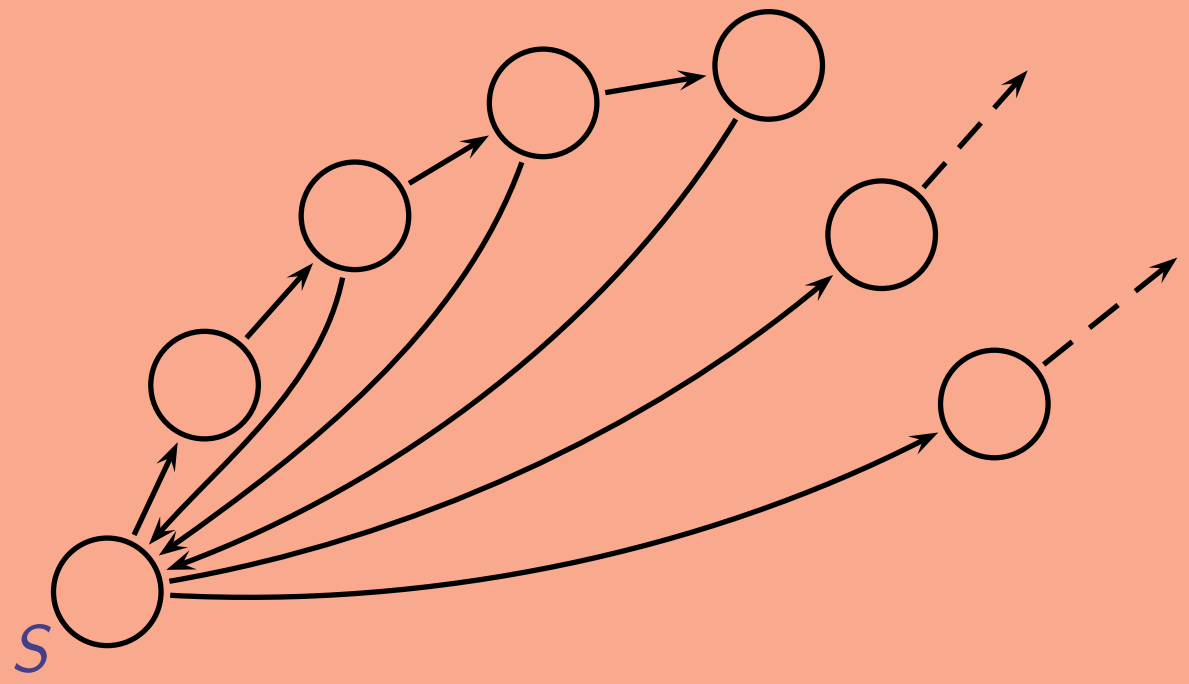
Wenn es möglich ist, von einem Zustand in einen anderen zu wechseln, zeichnen wir einen Pfeil dazwischen:



Beispiel 3: Handy

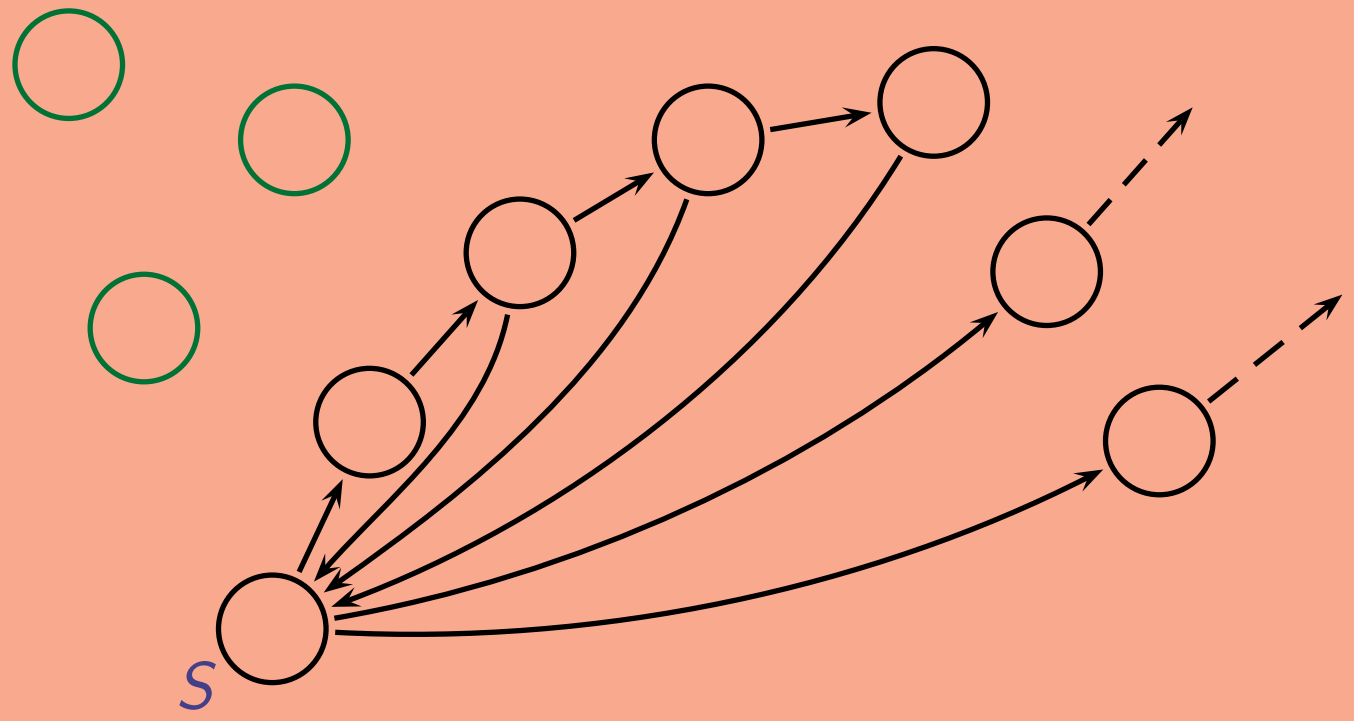
Wir stellen die Zustände durch Kreise dar.

Wenn es möglich ist, von einem Zustand in einen anderen zu wechseln, zeichnen wir einen Pfeil dazwischen:



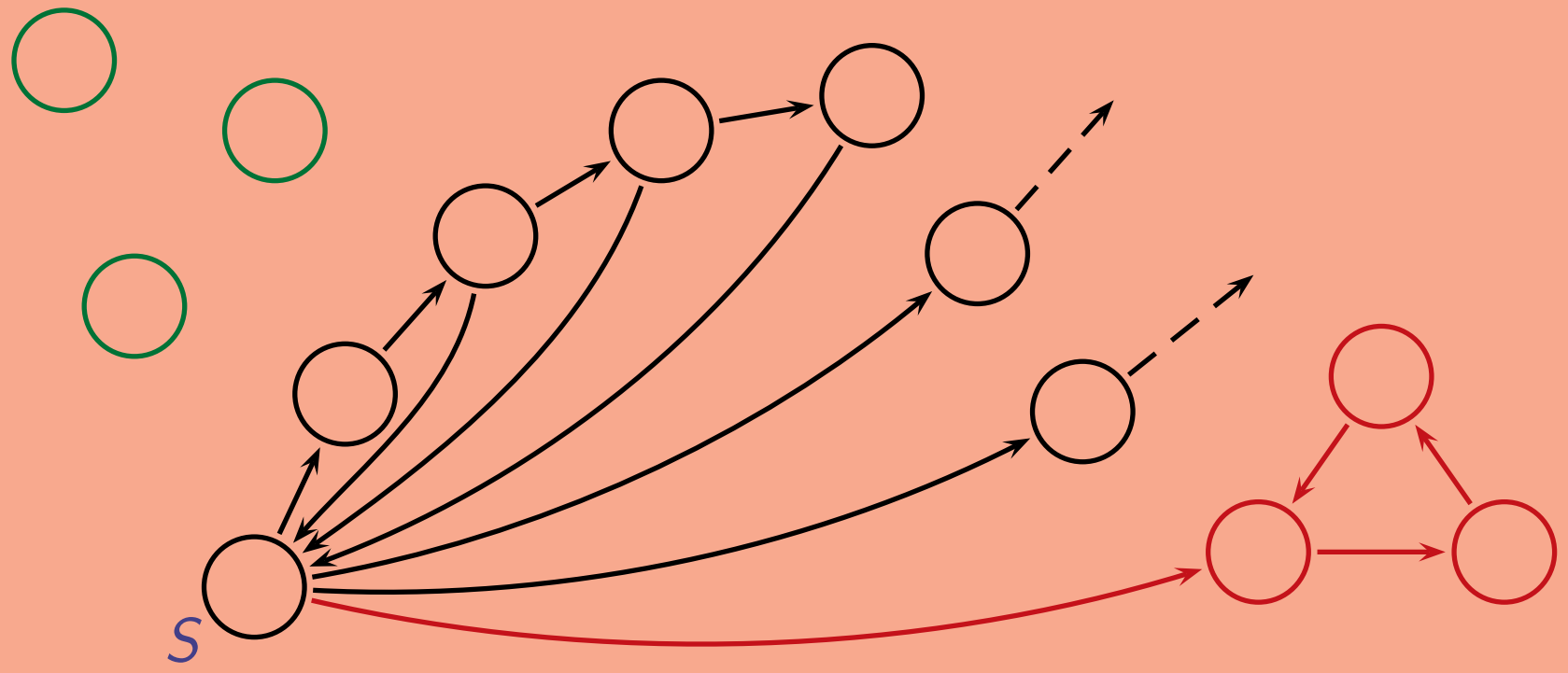
Beispiel 3: Handy

Es kann auch „unmögliche“ Zustände geben.
Aber diese dürfen dann natürlich nicht vom Startzustand aus erreichbar sein.



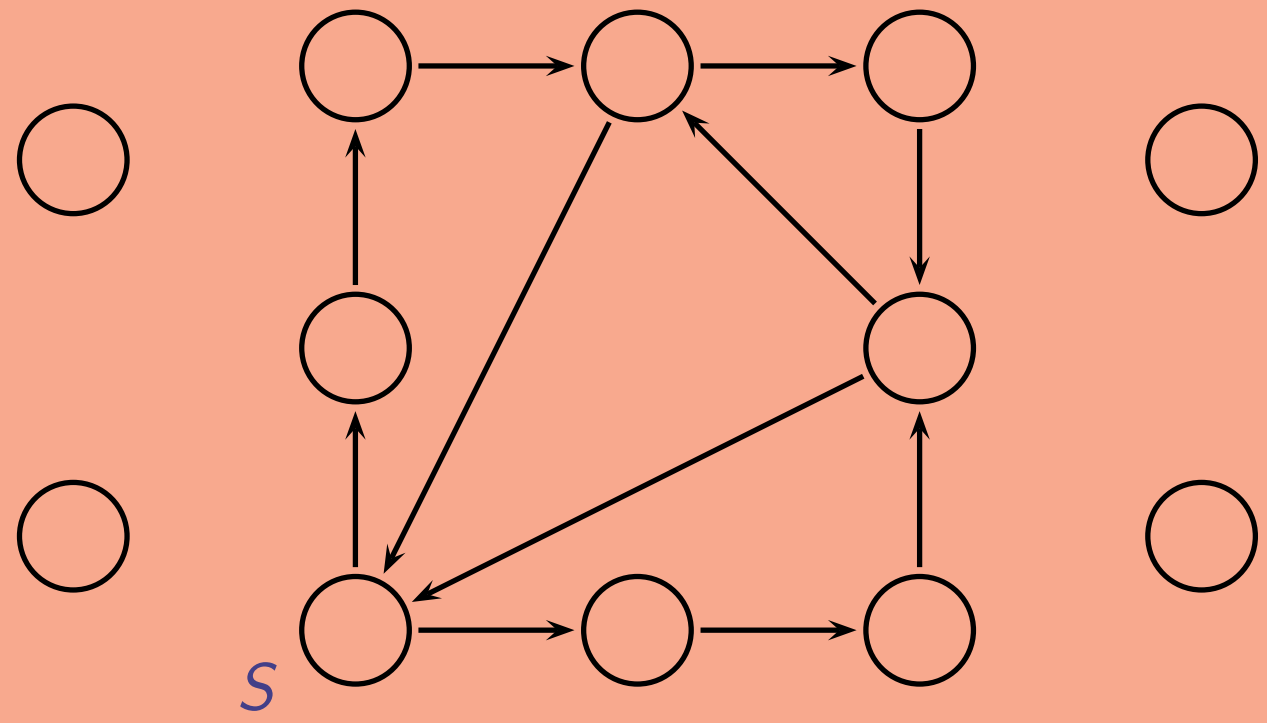
Beispiel 3: Handy

Was es aber auf keinen Fall geben darf, das sind Sackgassen.



Beispiel 3: Handy

Frage: Wie kann man automatisch feststellen, ob irgendein Zustandsdiagramm eine Sackgasse besitzt?

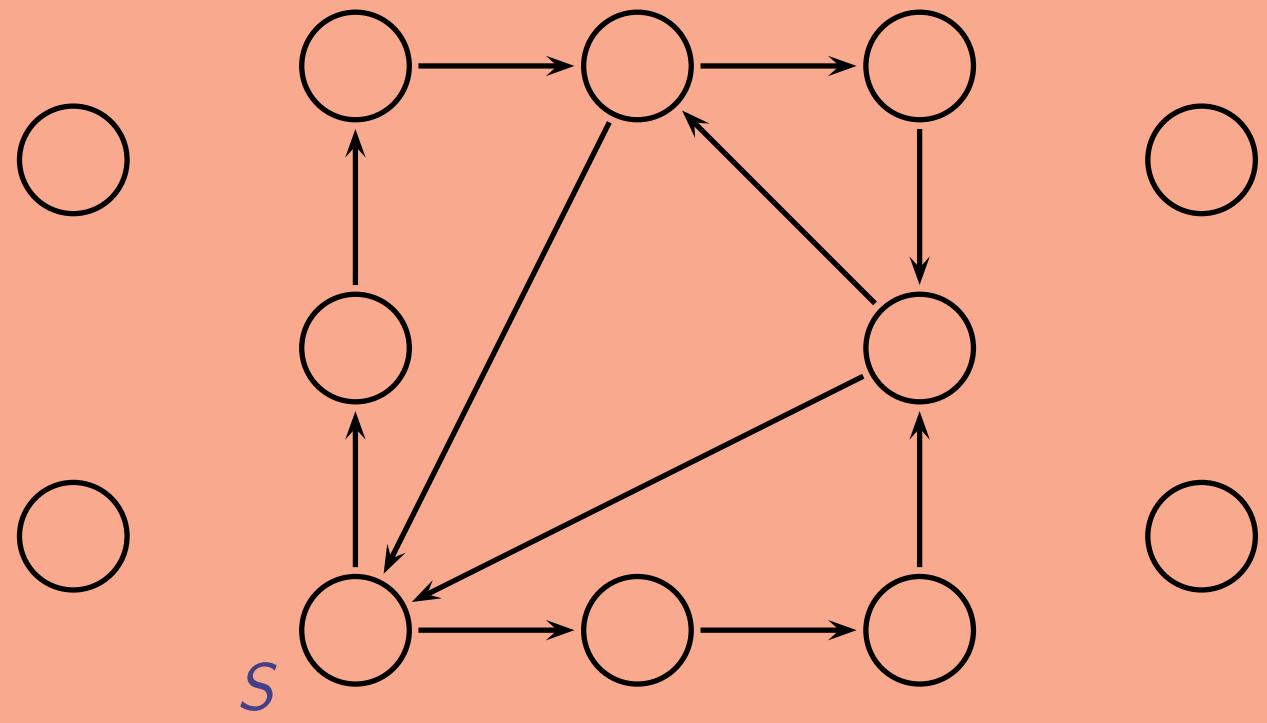


Beispiel 3: Handy

Als Formel ausgedrückt: Hat S die Eigenschaft $\mathsf{EF}\,\neg\,\mathsf{EF}\,S$?

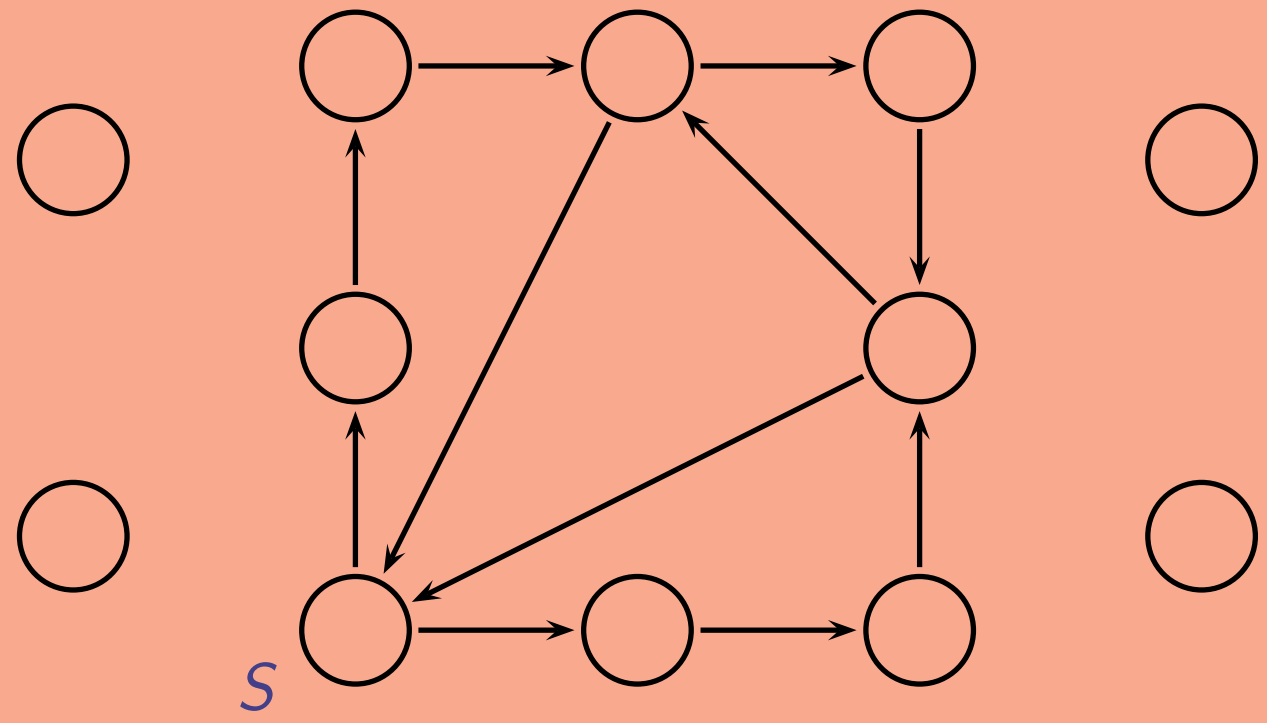
Dabei bedeutet EF:

„man kann einen Zustand erreichen mit der Eigenschaft . . . “



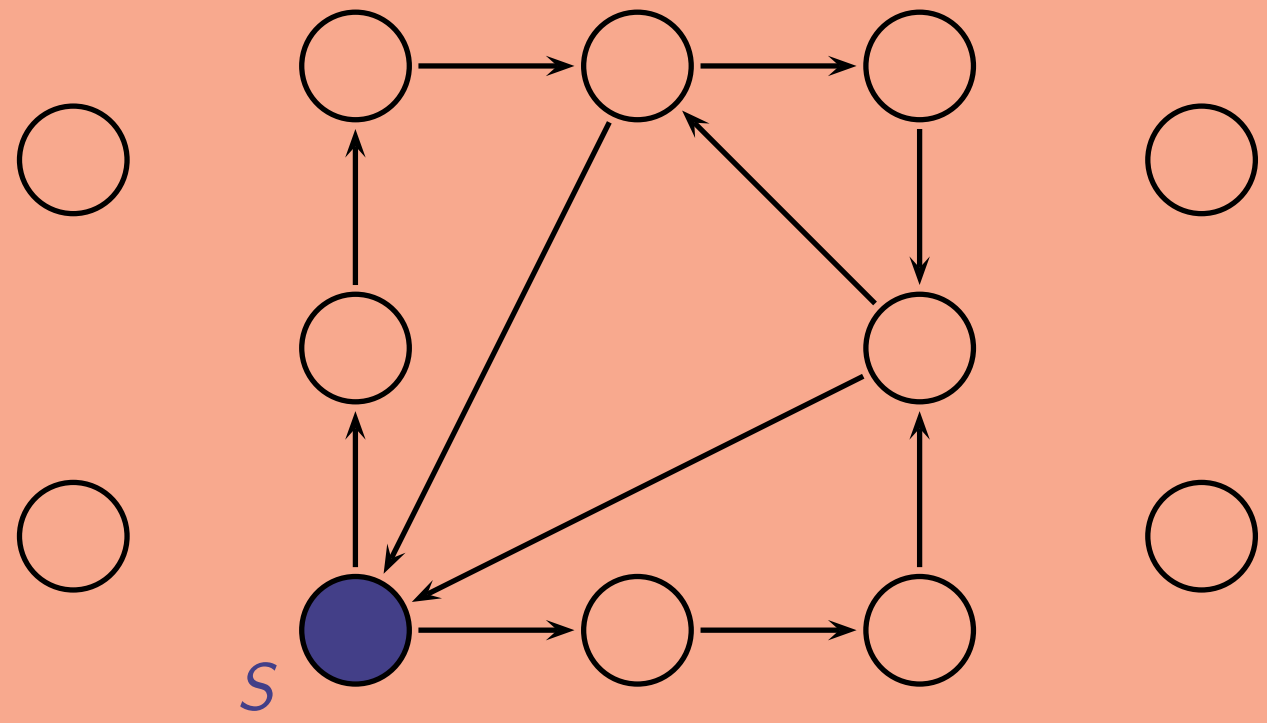
Beispiel 3: Handy

Zuerst ermitteln wir alle Zustände mit der Eigenschaft EF S,
das heißt, alle Zustände, von denen aus man S erreichen kann.



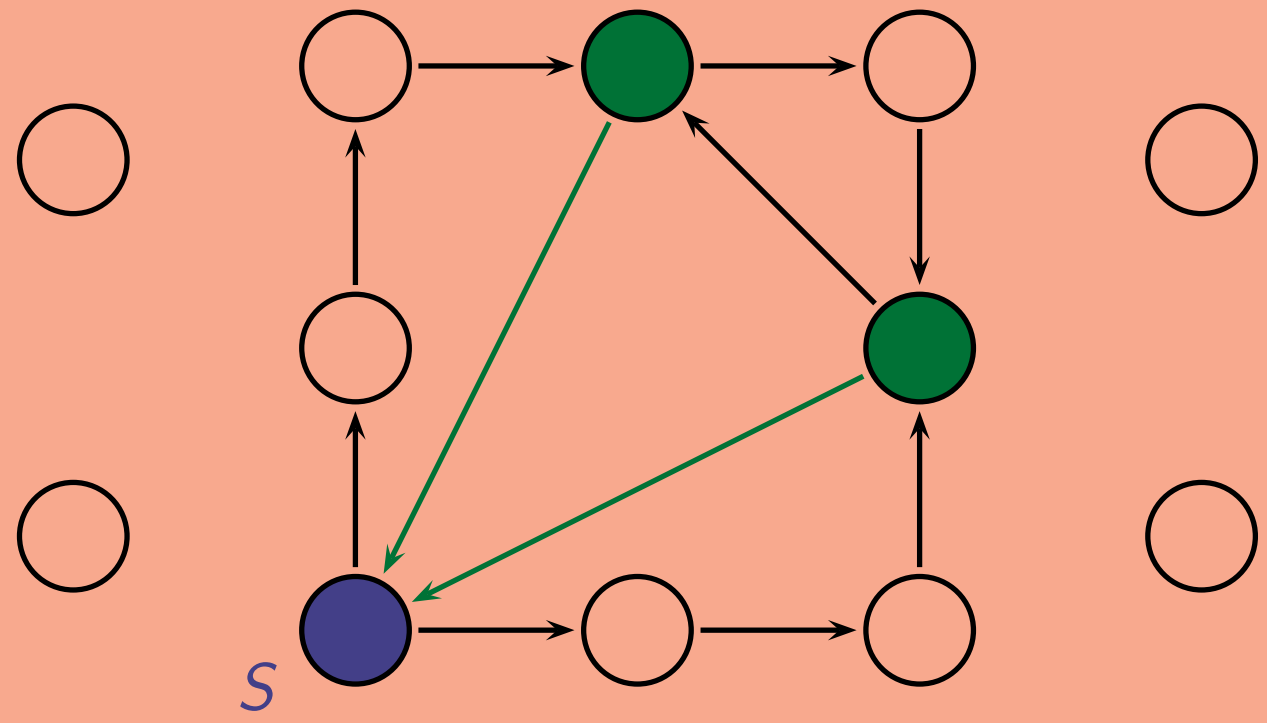
Beispiel 3: Handy

Zuerst ermitteln wir alle Zustände mit der Eigenschaft EF S,
das heißt, alle Zustände, von denen aus man S erreichen kann.



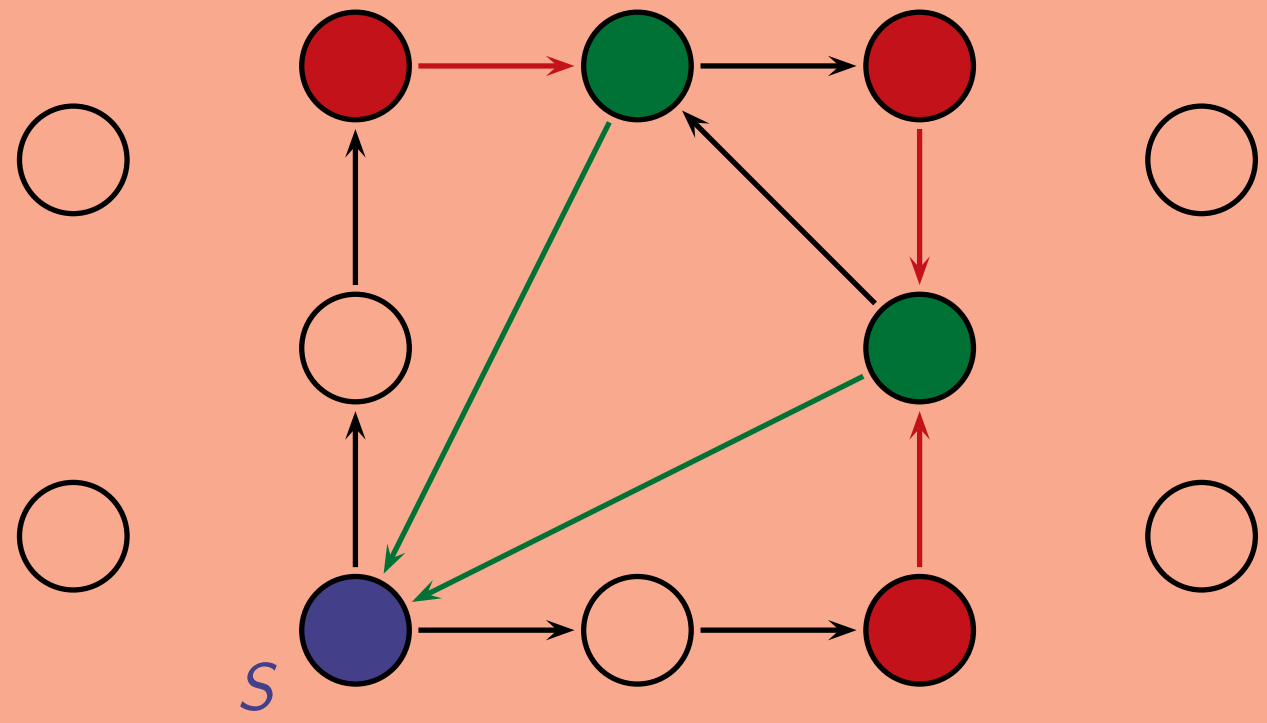
Beispiel 3: Handy

Zuerst ermitteln wir alle Zustände mit der Eigenschaft EF S,
das heißt, alle Zustände, von denen aus man S erreichen kann.



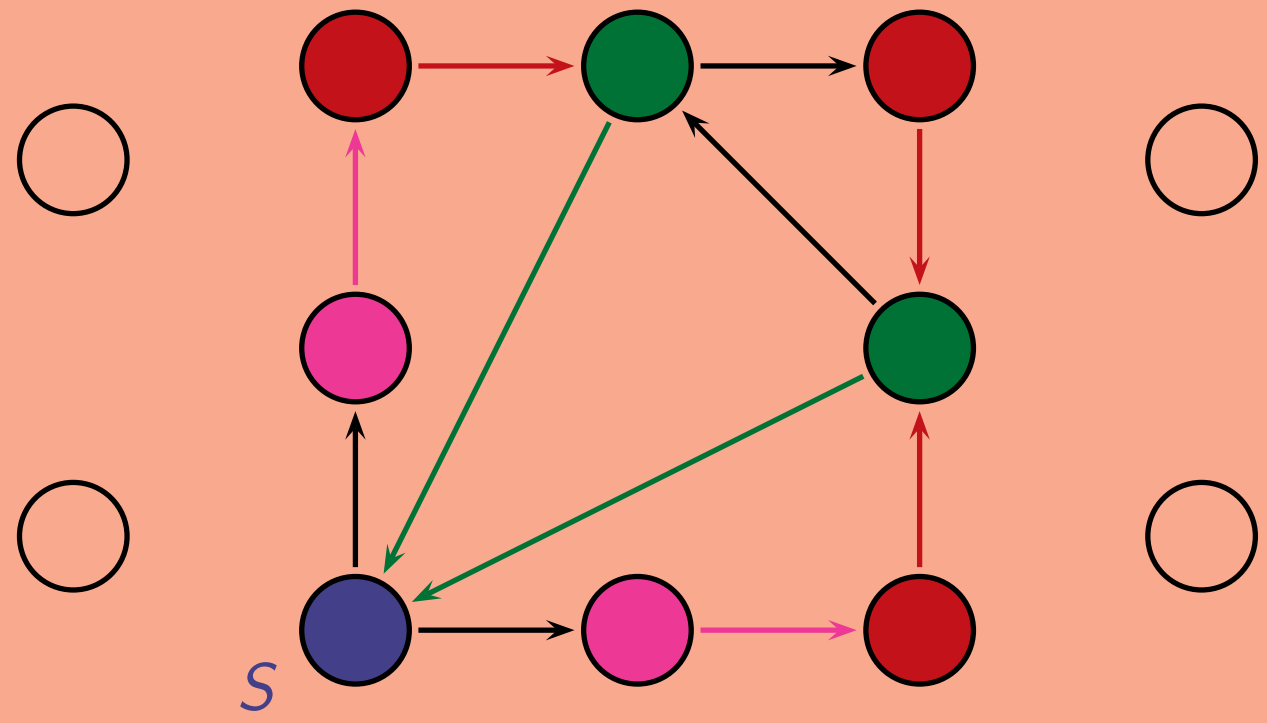
Beispiel 3: Handy

Zuerst ermitteln wir alle Zustände mit der Eigenschaft EF S,
das heißt, alle Zustände, von denen aus man S erreichen kann.



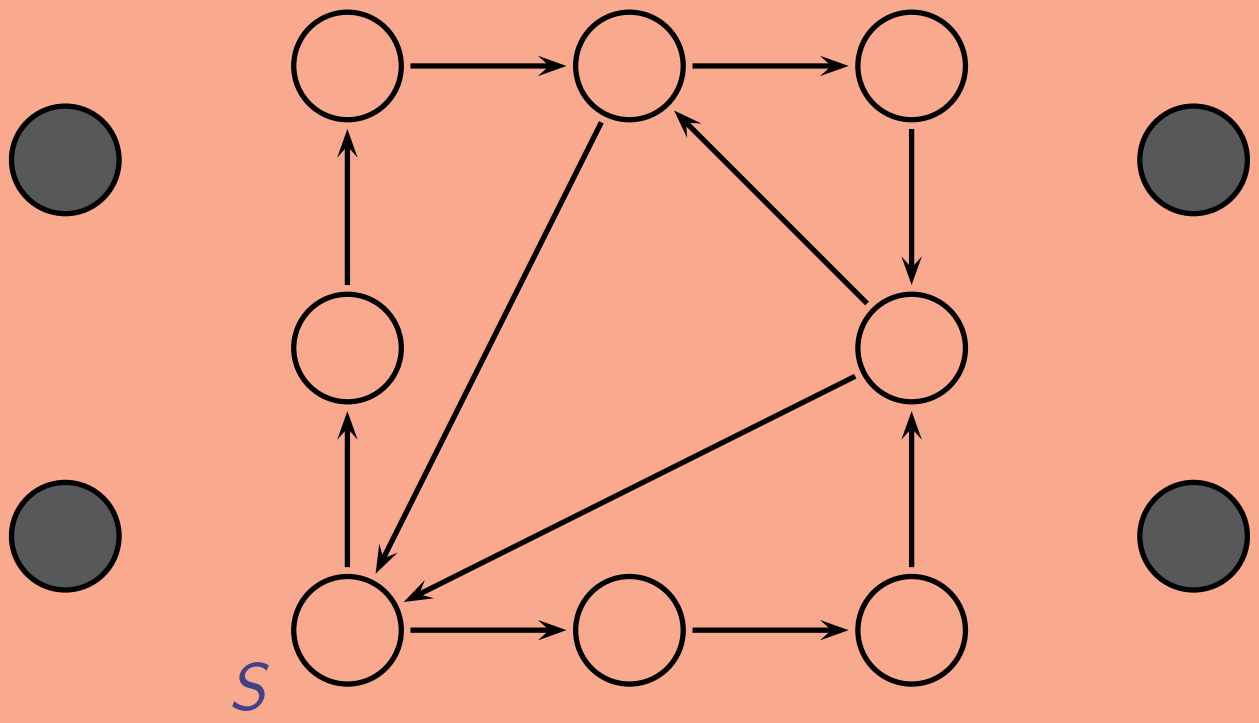
Beispiel 3: Handy

Zuerst ermitteln wir alle Zustände mit der Eigenschaft EF S,
das heißt, alle Zustände, von denen aus man S erreichen kann.



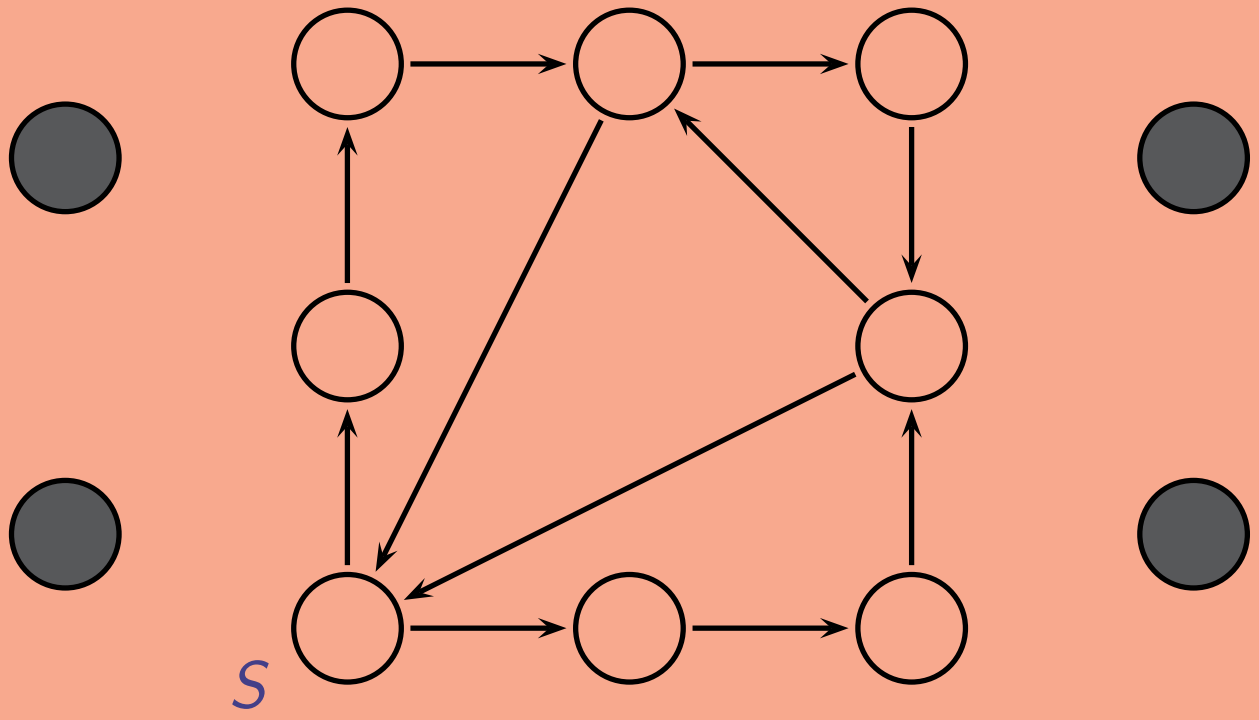
Beispiel 3: Handy

Dann ermitteln wir alle Zustände mit der Eigenschaft $\neg\ \mathrm{EF}\ S$, das heißt, alle Zustände, die **nicht** die Eigenschaft $\mathrm{EF}\ S$ haben.



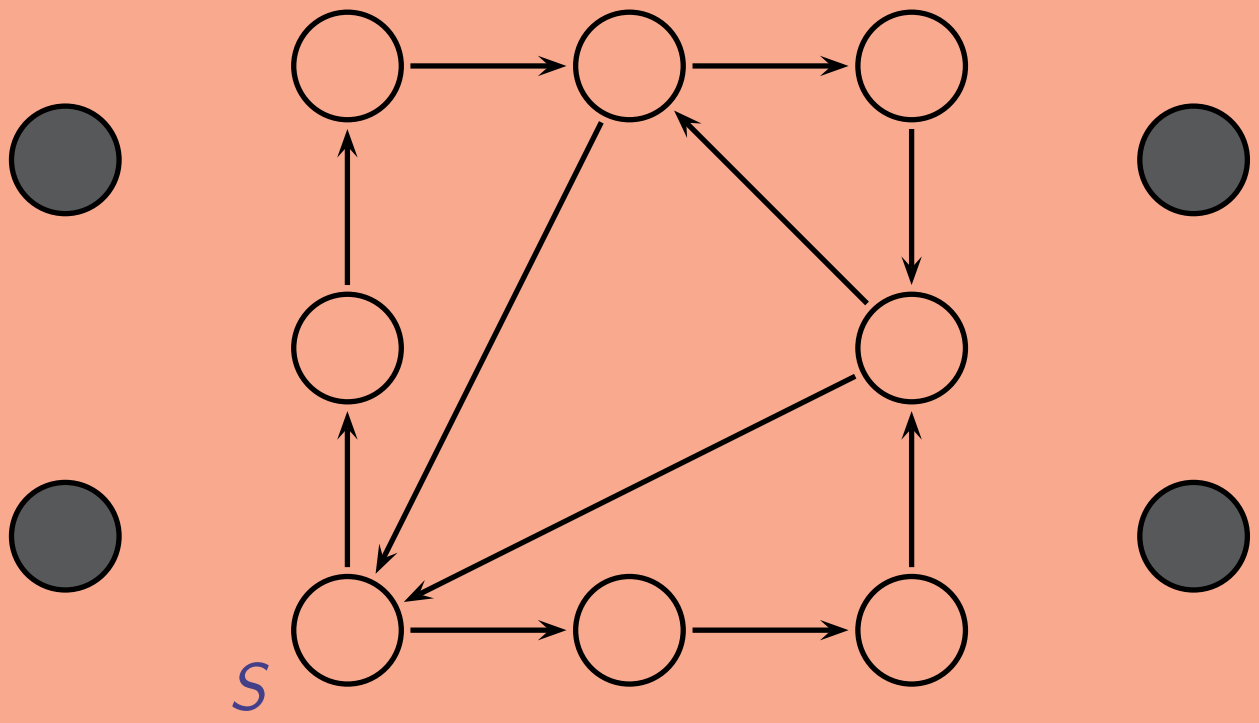
Beispiel 3: Handy

Zum Schluß ermitteln wir alle Zustände mit der Eigenschaft $\mathsf{EF}\,\neg\,\mathsf{EF}\,S$, also diejenigen, von denen aus man einen grauen Zustand erreicht. (Das sind aber gerade die grauen Zustände selbst.)



Beispiel 3: Handy

Da der Zustand S **nicht** zu diesen Zuständen gehört, wissen wir nun, daß wir von S aus keine Sackgasse erreichen.



Modellüberprüfung

Modellüberprüfung ist heute für jeden Chipentwickler eine Standardmethode.

Die Hauptschwierigkeit ist dabei die Größe der Probleme:

- 10 bits $\hat{=}$ 1 024 Zustände
- 20 bits $\hat{=}$ 1 048 576 Zustände
- 30 bits $\hat{=}$ 1 073 741 824 Zustände
- 40 bits $\hat{=}$ 1 099 511 627 776 Zustände

Modellüberprüfung

Modellüberprüfung ist heute für jeden Chipentwickler eine Standardmethode.

Die Hauptschwierigkeit ist dabei die Größe der Probleme:

- 10 bits $\hat{=}$ 1 024 Zustände
- 20 bits $\hat{=}$ 1 048 576 Zustände
- 30 bits $\hat{=}$ 1 073 741 824 Zustände
- 40 bits $\hat{=}$ 1 099 511 627 776 Zustände

Ausweg:

- Zustände und Übergange nicht explizit darstellen, sondern Zustandsmengen durch (hoffentlich kleine) Formeln kodieren.

Modellüberprüfung

Noch kniffliger:

- Software-Verifikation
- Hybride Systeme (d. h., Systeme, in denen Zustände auch durch reelle Zahlen beschrieben werden)

In beiden Fällen ist die Anzahl der Zustände unendlich.

Modellüberprüfung

Noch kniffliger:

- Software-Verifikation
- Hybride Systeme (d. h., Systeme, in denen Zustände auch durch reelle Zahlen beschrieben werden)

In beiden Fällen ist die Anzahl der Zustände unendlich.

Ausweg:

Zustände zu endlich vielen Intervallen oder Klassen zusammenfassen (funktioniert aber leider nicht immer).

Softwareverifikation

$T = (S, \rightarrow)$, $\quad L : S \rightarrow \mathcal{P}(\Pi)$, wobei $\Pi = \{p_1, \ldots, p_n\}$

- S Menge von Zustände
- $\rightarrow$ Transitionsrelation
- $L(s)$ Menge aller Aussagenvariablen, die in Zustand s wahr sind.

Softwareverifikation

$T = (S, \rightarrow)$, $\quad L : S \rightarrow \mathcal{P}(\Pi)$, wobei $\Pi = \{p_1, \ldots, p_n\}$

Kodierung als Formeln

- **(Mengen von) Elementen von S:** Formeln

  $s \in S \mapsto F_s := L_1 \wedge \cdots \wedge L_n$, wobei

  $L_i = p_i$ falls $p_i \in L(s)$ (d.h.. falls p_i wahr in s)

  $L_i = \neg p_i$ falls $p_i \notin L(s)$ (d.h. falls $\neg p_i$ wahr in s)

  (Konjunktion aller Literale, die in Zustand s wahr sind)

  $U = \{s_1, \ldots, s_m\} \subseteq S \mapsto F_U = F_{s_1} \vee \cdots \vee F_{s_n}$

Softwareverifikation

$T = (S, \rightarrow)$, $\quad L : S \rightarrow \mathcal{P}(\Pi)$, wobei $\Pi = \{p_1, \ldots, p_n\}$

Kodierung als Formeln

- **Transitionsrelation $\rightarrow$:** Formel $F_\rightarrow$

  Kopie der Menge der Aussagenvariablen $\Pi' = \{p'_1, \ldots, p'_n\}$.

$$s \rightarrow s' \mapsto F_{s \rightarrow s'} = \underbrace{(L_1 \wedge L_2 \wedge \cdots \wedge L_n)}_{F_s(p_1, \ldots, p_n)} \wedge \underbrace{(L'_1 \wedge L'_2 \wedge \cdots \wedge L'_n)}_{F_{s'}(p'_1, \ldots, p'_n)}$$

Eine Transition von s nach s':

zuerst in s, d.h. alle Literale, die in s wahr sind sind wahr $(F_s(p_1, \ldots, p_n))$

Nach Transition neue Werte für die Aussagenvariablen (Änderung $p_i \mapsto p'_i$) . Da nach Transition in Zustand s', alle Literale, die in s' wahr sind, sind wahr nach der Änderung $(F_{s'}(p'_1, \ldots, p'_n))$

$F_\rightarrow = \bigvee_{s \rightarrow s'} F_{s \rightarrow s'}$

Softwareverifikation

Erreichbare Zustände: Formel (kann berechnet werden) F_e

"Schlechte" Zustände: F_{bad}.

Können "schlechte" Zustände erreicht werden?

Ist $F_e \wedge F_{\text{bad}}$ erfüllbar?

Mehr in der Vorlesung "Formale Spezifikation und Verifikation"

Beispiel

Frage: Sortiert BubbleSort die Eingabe?

```
int [] BubbleSort(int[] a) {
    int i, j, t;
    for (i := |a| − 1; i > 0; i := i − 1) {
        for (j := 0; j < i; j := j + 1) {
            if (a[j] > a[j + 1]){t := a[j];
                                 a[j] := a[j + 1];
                                 a[j + 1] := t};
}} return a}
```

Kann nicht zu einem Erfüllbarkeitstest in Aussagenlogik reduziert werden.

Beispiel

Frage: Sortiert BubbleSort die Eingabe?

```
int [] BubbleSort(int[] a) {
    int i, j, t;
    for (i := |a| − 1; i > 0; i := i − 1) {
        for (j := 0; j < i; j := j + 1) {
            if (a[j] > a[j + 1]){t := a[j];
                                 a[j] := a[j + 1];
                                 a[j + 1] := t};
}} return a}
```

Einfachere Frage:

$|a| = 3$; $a[0]{=}7, a[1]{=}9, a[2]{=}4$

Sortiert BubbleSort diese Eingabe?

Kodierung in Aussagenlogik:

- p_{ij}^k (in Schritt k, $a[i] = j$)

  Beispiele: $p_{07}^1, p_{19}^1, p_{24}^1$
- gt_{ij}^k (in Schritt k, $a[i] > a[j]$)

  Beispiel: $gt_{10}^1, \neg gt_{01}^1, gt_{02}^1, \neg gt_{20}^1, \ldots$

Transitionen $\mapsto$ neue Aussagenvariablen wie vorher.

(nicht sehr expressiv!)

Anwendungsbereiche

1. Logische Schaltkreise
2. Puzzles
3. Planung
4. Verifikation