

Logik für Informatiker

2. Aussagenlogik

Teil 8

17.05.2018

Viorica Sofronie-Stokkermans

Universität Koblenz-Landau

e-mail: sofronie@uni-koblenz.de

Bis jetzt

- Normalformen: CNF/DNF
- Subsumption
- SAT-Problem (Erfüllbarkeitsproblem)
 - SAT; k -SAT
 - 3-SAT vs. SAT (NP-complete)
 - 2-SAT: PTIME
 - Horn-Formeln (Erfüllbarkeitstest) PTIME
- Der aussagenlogische Resolutionkalkül
- Semantische Tableaux

Der aussagenlogische Tableaukalkül

Wesentliche Eigenschaften

- Widerlegungskalkül: Testet auf Unerfüllbarkeit
- Beweis durch Fallunterscheidung
- Top-down-Analyse der gegebenen Formeln

Vorteile

- Intuitiver als Resolution
- Formeln müssen nicht in Normalform sein
- Falls Formelmenge erfüllbar ist (Test schlägt fehl), wird ein Gegenbeispiel (eine erfüllende Interpretation) konstruiert

Nachteile

- Mehr als eine Regel

Formeltypen

Konjunktive Formeln: Typ α

- $\neg\neg F$
- $F \wedge G$
- $\neg(F \vee G)$
- $\neg(F \rightarrow G)$

Zuordnungsregeln Formeln / Unterformeln

α	α_1	α_2
$F \wedge G$	F	G
$\neg(F \vee G)$	$\neg F$	$\neg G$
$\neg(F \rightarrow G)$	F	$\neg G$
$\neg\neg F$	F	

Formeltypen

Disjunktive Formeln: Typ β

- $\neg(F \wedge G)$
- $F \vee G$
- $F \rightarrow G$

Zuordnungsregeln Formeln / Unterformeln

β	β_1	β_2
$\neg(F \wedge G)$	$\neg F$	$\neg G$
$F \vee G$	F	G
$F \rightarrow G$	$\neg F$	G

Regeln des (aussagenlogischen) Tableauealküls

α			$p \wedge q$
<hr/>			
α_1	Konjunktiv		p
α_2			q
β			$p \vee q$
<hr/>			/ \
β_1 β_2	Disjunktiv		p q
ϕ			ϕ
$\neg\phi$	Widerspruch		$\neg\phi$
<hr/>			
\perp			\perp

Determinismus von Kalkül und Regeln

Determinismus

- Die Regeln sind alle deterministisch
- Der Kalkül aber nicht:
Auswahl der nächsten Formel, auf die eine Regel angewendet wird

Heuristik

- Nicht-verzweigende Regeln zuerst: α vor β

Nota bene:

Dieselbe Formel kann mehrfach (auf verschiedenen Ästen) verwendet werden

Klauseltableau

M Menge von Klauseln

Änderungen

- Keine α -Regel
- Erweiterungsregel kann Verzweigungsgrad > 2 haben
- Alle Knoten im Tableau enthalten Literale

Korrektheit und Vollständigkeit des Tableaukalküls

Theorem.

Eine Formelmenge M ist unerfüllbar
genau dann, wenn
es einen Tableaubeweis für (die Unerfüllbarkeit von) M gibt

Korrektheit: “ \Leftarrow ”

Falls es einen Tableaubeweis für (die Unerfüllbarkeit von) M gibt
(d.h. ein Tableau für M , das geschlossen ist) so ist M unerfüllbar.

... **alternativ:** Falls M erfüllbar ist, hat M kein geschlossenes Tableau.

Vollständigkeit: “ \Rightarrow ”

Falls M unerfüllbar ist, gibt es einen Tableaubeweis für (die Unerfüllbarkeit von) M
(d.h. ein Tableau für M , das geschlossen ist).

Klauseltableau: Einschränkungen des Suchraums

Regularität:

Kein Literal darf auf einem Ast mehr als einmal vorkommen

Schwache Konnektionsbedingung (Connection calculus)

Bei Erweiterung von Ast B muss mindestens eines der neuen Literale komplementär zu Literal in B sein.

Starke Konnektionsbedingung (Modellelimination)

Bei Erweiterung von Ast B muss mindestens eines der neuen Literale komplementär zum **Blatt** von B sein – außer beim ersten Schritt.

Klauseltableau: Einschränkungen des Suchraums

Theorem (hier ohne Beweis)

Regularität, starke und schwache Konnektionsbedingung
erhalten Vollständigkeit.

Falls eine Formelmengemenge M unerfüllbar ist, so gibt es ein Tableau für M , das geschlossen ist und die Regularitätsbedingung (bzw. starke Konnektionsbedingung, bzw. schwache Konnektionsbedingung) erfüllt.

Klauseltableau: Einschränkungen des Suchraums

Theorem (hier ohne Beweis)

Regularität, starke und schwache Konnektionsbedingung
erhalten Vollständigkeit.

Falls eine Formelmenge M unerfüllbar ist, so gibt es ein Tableau für M , das geschlossen ist und die Regularitätsbedingung (bzw. starke Konnektionsbedingung, bzw. schwache Konnektionsbedingung) erfüllt.

Jedoch

Bei starker Konnektionsbedingung kann ungünstige Erweiterung in Sackgasse führen.

(bei schwacher Konnektionsbedingung nicht)

Klauseltableau: Weiteres Beispiel

Signatur: *F*: Flugreise *V*: Vollpension *M*: Meer *P*: Pool

Klauseltableau: Weiteres Beispiel

Signatur: F : Flugreise V : Vollpension M : Meer P : Pool

Falls sie nicht mit dem Flugzeug fliegen, besteht der Vater auf Vollpension am Meer.

$$\neg F \rightarrow (V \wedge M)$$

Klauseltableau: Weiteres Beispiel

Signatur: F : Flugreise V : Vollpension M : Meer P : Pool

Falls sie nicht mit dem Flugzeug fliegen, besteht der Vater auf Vollpension am Meer.

$$\neg F \rightarrow (V \wedge M)$$

Die Mutter möchte mindestens einen ihrer drei Wünsche erfüllt sehen: ans Meer fliegen, oder am Meer ohne Pool, oder Vollpension und Pool.

$$(M \wedge F) \vee (M \wedge \neg P) \vee (V \wedge P)$$

Klauseltableau: Weiteres Beispiel

Signatur: F : Flugreise V : Vollpension M : Meer P : Pool

Falls sie nicht mit dem Flugzeug fliegen, besteht der Vater auf Vollpension am Meer.

$$\neg F \rightarrow (V \wedge M)$$

Die Mutter möchte mindestens einen ihrer drei Wünsche erfüllt sehen: ans Meer fliegen, oder am Meer ohne Pool, oder Vollpension und Pool.

$$(M \wedge F) \vee (M \wedge \neg P) \vee (V \wedge P)$$

Gibt es keinen Pool, so besteht Tochter Lisa auf einer Flugreise und Urlaub am Meer und darauf, dass keine Vollpension gebucht wird.

$$\neg P \rightarrow (F \wedge M \wedge \neg V)$$

Klauseltableau: Weiteres Beispiel

Signatur: F : Flugreise V : Vollpension M : Meer P : Pool

Falls sie nicht mit dem Flugzeug fliegen, besteht der Vater auf Vollpension am Meer.

$$\neg F \rightarrow (V \wedge M)$$

Die Mutter möchte mindestens einen ihrer drei Wünsche erfüllt sehen: ans Meer fliegen, oder am Meer ohne Pool, oder Vollpension und Pool.

$$(M \wedge F) \vee (M \wedge \neg P) \vee (V \wedge P)$$

Gibt es keinen Pool, so besteht Tochter Lisa auf einer Flugreise und Urlaub am Meer und darauf, dass keine Vollpension gebucht wird.

$$\neg P \rightarrow (F \wedge M \wedge \neg V)$$

Auch dem Baby soll einer seiner Wünsche erfüllt werden: erstens einen Pool und nicht fliegen oder zweitens Vollpension, dann aber ohne Pool.

$$(P \wedge \neg F) \vee (V \wedge \neg P)$$

Klauseltableau: Weiteres Beispiel

Behauptung

Dann müssen sie ans Meer mit Vollpension, mit Pool und ohne Flug.

$$M \wedge V \wedge P \wedge \neg F$$

Klauseltableau: Weiteres Beispiel

Behauptung

Dann müssen sie ans Meer mit Vollpension, mit Pool und ohne Flug.

$$M \wedge V \wedge P \wedge \neg F$$

Zu zeigen:

$$\{\neg F \rightarrow (V \wedge M), (M \wedge F) \vee (M \wedge \neg P) \vee (V \wedge P), \\ \neg P \rightarrow (F \wedge M \wedge \neg V), (P \wedge \neg F) \vee (V \wedge \neg P)\} \models M \wedge V \wedge P \wedge \neg F$$

Klauseltableau: Weiteres Beispiel

Behauptung

Dann müssen sie ans Meer mit Vollpension, mit Pool und ohne Flug.

$$M \wedge V \wedge P \wedge \neg F$$

Zu zeigen:

$$\{\neg F \rightarrow (V \wedge M), (M \wedge F) \vee (M \wedge \neg P) \vee (V \wedge P), \\ \neg P \rightarrow (F \wedge M \wedge \neg V), (P \wedge \neg F) \vee (V \wedge \neg P)\} \models M \wedge V \wedge P \wedge \neg F$$

Wir zeigen, dass die folgende Konjunktion unerfüllbar ist:

$$\begin{aligned} &(\neg F \rightarrow (V \wedge M), (M \wedge F)) \wedge \\ &((M \wedge F) \vee (M \wedge \neg P) \vee (V \wedge P)) \wedge \\ &(\neg P \rightarrow (F \wedge M \wedge \neg V)) \wedge \\ &((P \wedge \neg F) \vee (V \wedge \neg P)) \wedge \\ &\neg M \vee \neg V \vee \neg P \vee F \end{aligned}$$

(Negation der Behauptung)

Klauseltableau: Weiteres Beispiel

$$\neg F \rightarrow (V \wedge M)$$

$$(M \wedge F) \vee (M \wedge \neg P) \vee (V \wedge P)$$

$$\neg P \rightarrow (F \wedge M \wedge \neg V)$$

$$(P \wedge \neg F) \vee (V \wedge \neg P)$$

Negation der Behauptung

- | | |
|------|---|
| (1) | $F \vee V$ |
| (2) | $F \vee M$ |
| (3) | $M \vee V$ |
| (4) | $M \vee P$ |
| (5) | $M \vee \neg P \vee V$ |
| (6) | $F \vee M \vee V$ |
| (7) | $F \vee M \vee P$ |
| (8) | $F \vee \neg P \vee V$ |
| (9) | $P \vee F$ |
| (10) | $P \vee M$ |
| (11) | $P \vee \neg V$ |
| (12) | $P \vee V$ |
| (13) | $\neg F \vee V$ |
| (14) | $\neg F \vee \neg P$ |
| (15) | $\neg M \vee \neg V \vee \neg P \vee F$ |

Klauseltableau: Weiteres Beispiel

Beobachtung

Konstruktion des Konnektionstableaus

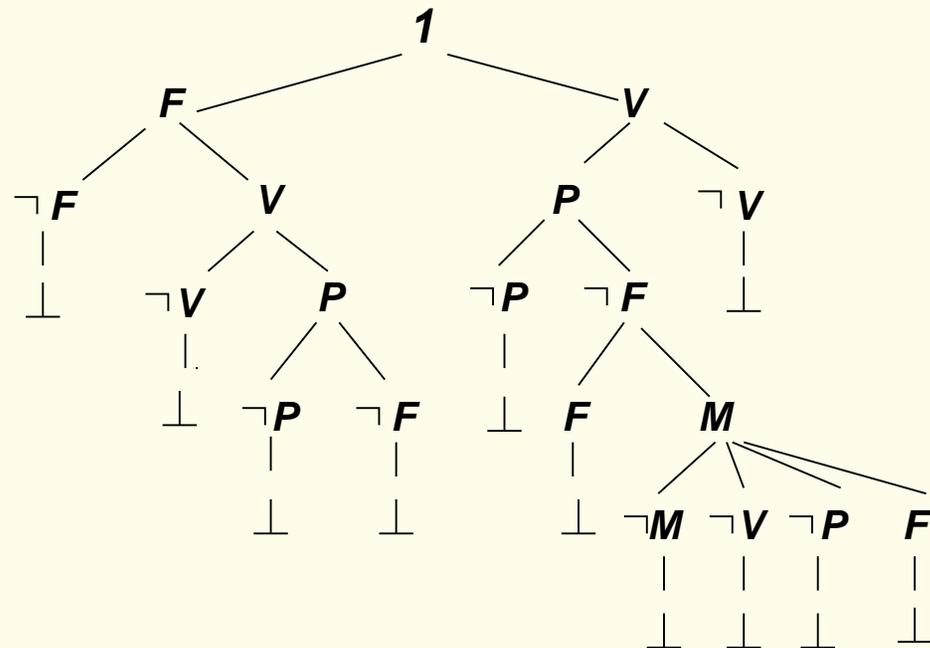
- bei Beginn mit Klausel (1)
- mit Regularität
- mit starker Konnektionsbedingung

Dann:

Nahezu deterministische Beweiskonstruktion

Klauseltableau: Weiteres Beispiel

- (1) $F \vee V$
- (2) $F \vee M$
- (3) $M \vee V$
- (4) $M \vee P$
- (5) $M \vee \neg P \vee V$
- (6) $F \vee M \vee V$
- (7) $F \vee M \vee P$
- (8) $F \vee \neg P \vee V$
- (9) $P \vee F$
- (10) $P \vee M$
- (11) $P \vee \neg V$
- (12) $P \vee V$
- (13) $\neg F \vee V$
- (14) $\neg F \vee \neg P$
- (15) $\neg M \vee \neg V \vee \neg P \vee F$



Terminierung

Resolution (für Klauselmengen in Mengennotation) terminiert, für jede endliche Menge von Klauseln.

Ähnliches Resultat für Tableaux?

Terminierung

Definition. Ein Tableau ist **strikt**, falls für jede Formel F die entsprechende Erweiterungsregel höchstens einmal auf jeden Ast, der die Formel enthält, angewandt wurde.

Theorem.

Sei T ein striktes aussagenlogisches Tableau. Dann ist T endlich.

Beweis: Neue Formeln mit denen ein Tableau erweitert wird sind \perp , (\top) oder Teilformeln der Formel, auf der die Erweiterungsregel angewandt wird. Da T strikt ist, wird für jede Formel F die entsprechende Erweiterungsregel höchstens einmal auf jeden Ast, der die Formel enthält, angewandt.

Dann sind alle Äste in T endlich, und so ist auch T endlich (König's Lemma).

Zusammenfassung: Tableaukalkül

Beweis durch Widerspruch und Fallunterscheidung

- Tableauregeln (mit uniformer Notation)
- Formale Definition des Kalküls
- Korrektheit und Vollständigkeit
- Klauseltableau
- Regularität
- Schwache und starke Konnektionsbedingung
- Strikte Tableaux
- Terminierung

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)

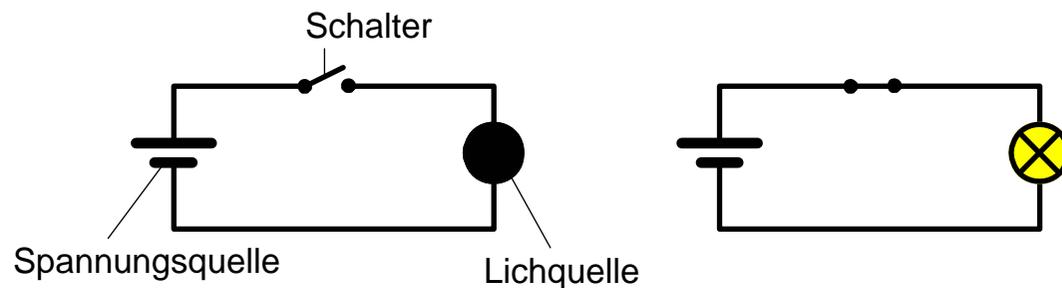
Schaltalgebra und kombinatorische Logik

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



Digitale elektrische Schaltungen

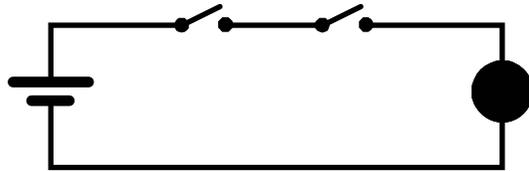
- eine einfache Schaltung:



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

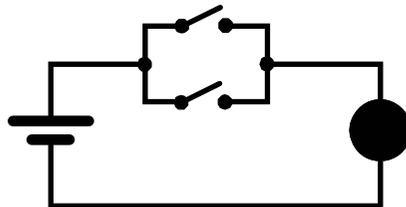
Digitale elektrische Schaltungen

- eine einfache **Reihenschaltung**:



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

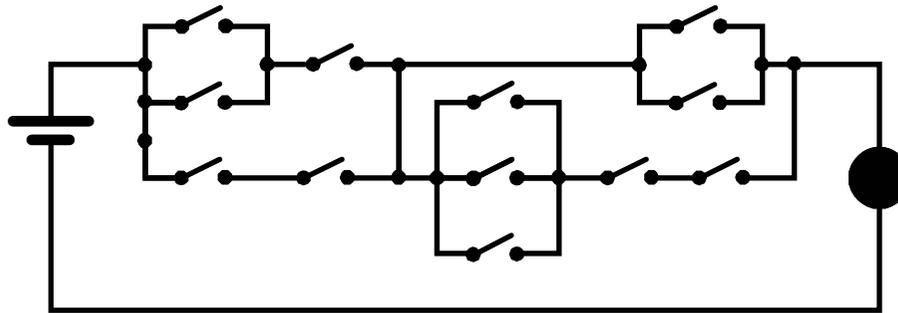
- eine einfache **Parallelschaltung**:



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

Digitale elektrische Schaltungen

- eine **komplexe(re) Schaltung**:



⇒ wann leuchtet die Lampe ?

⇒ 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 Boolesche Algebra zur Beschreibung von Schaltkreisen anwenden kann.



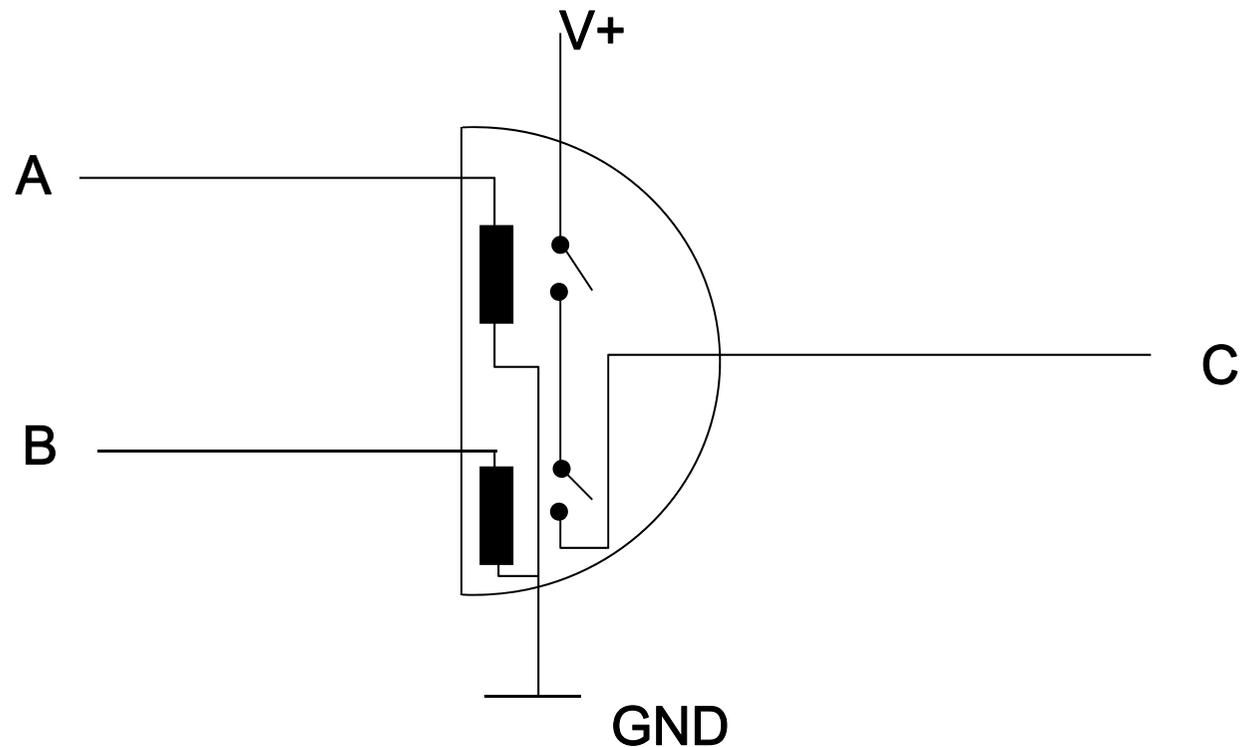
Bild: http://de.wikipedia.org/wiki/Claude_Shannon



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

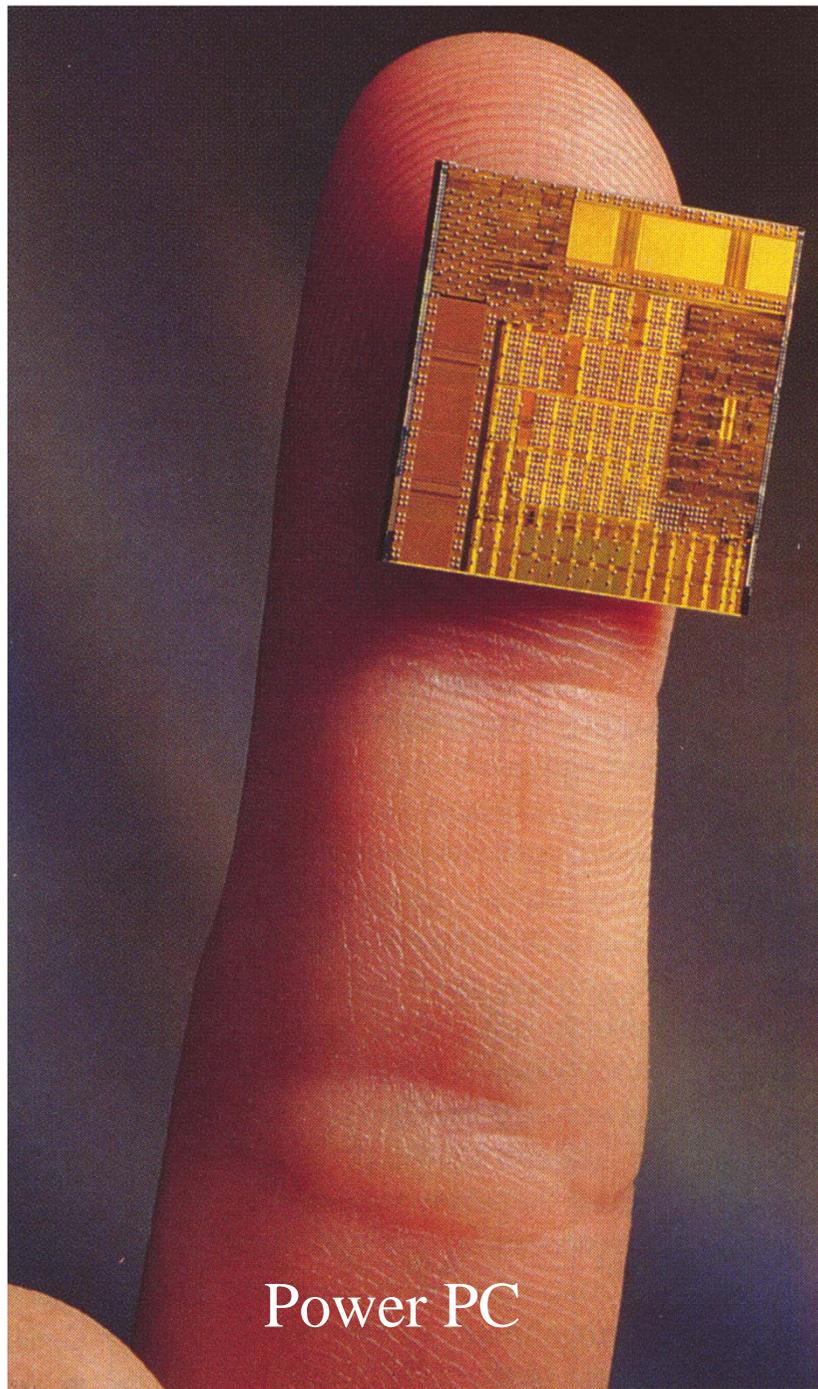
Realisierung einer UND-Funktion durch Gatter mit Relais:

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



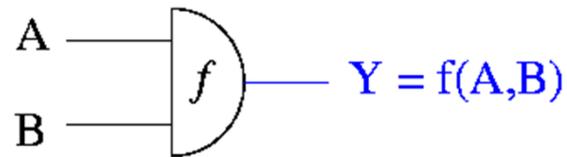


Realisierung durch
elektronische Schaltkreise:

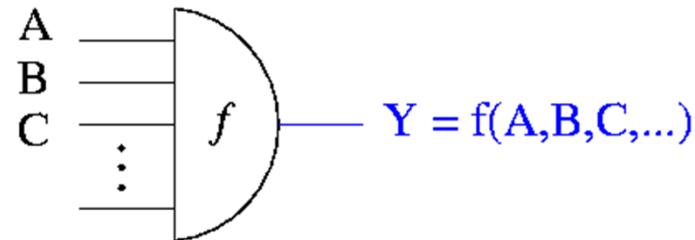
klein
sparsam
schnell
zuverlässig
billig

Logische Schaltungen

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



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:

$$Y = A_1 \cdot A_2 \cdot \dots \cdot A_k$$

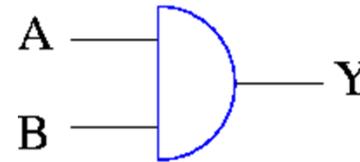
- Andere äquivalente Notationen:

$$Y = A \wedge B$$

$$Y = AB$$

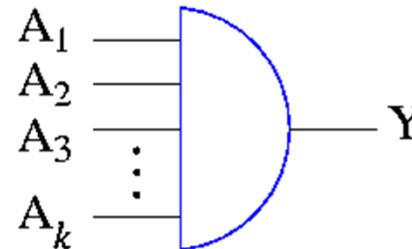
$$Y = A \& B$$

für 2 Eingänge:

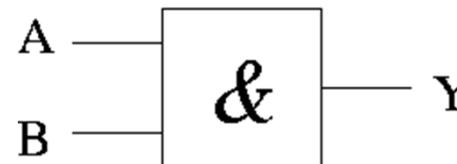


A	B	$Y = A \cdot B$
0	0	0
0	1	0
1	0	0
1	1	1

für k Eingänge:



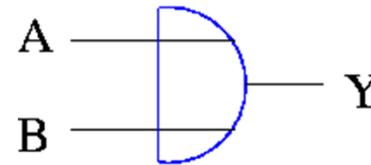
andere Darstellung:



Elementare Gatter

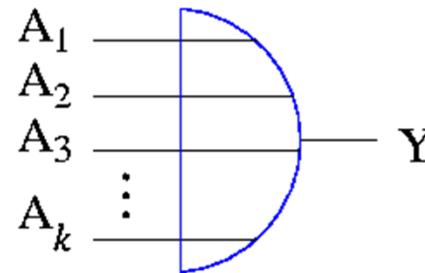
- ein **ODER**-Gatter realisiert eine **Disjunktion**:
 - für zwei Eingänge: $Y=A+B$,
 - für k Eingänge: $Y=A_1+A_2+\dots+A_k$

für 2 Eingänge:



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

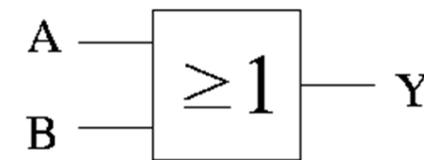
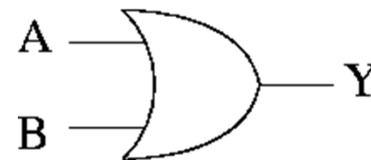
für k Eingänge:



- andere äquivalente Notationen:

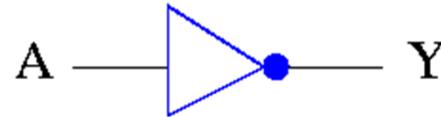
$$Y=A \vee B$$

andere Darstellungen:



Elementare Gatter

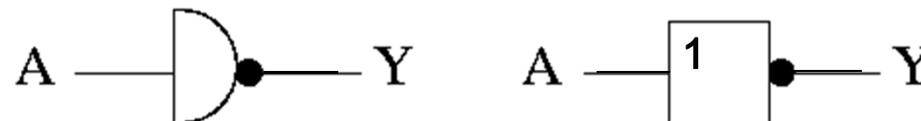
- ein **NICHT**-Gatter realisiert eine **Negation**:
 $Y = \overline{A}$



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

- andere äquivalente Notationen:
 $Y = A'$
 $Y = \neg A$

andere Darstellungen:



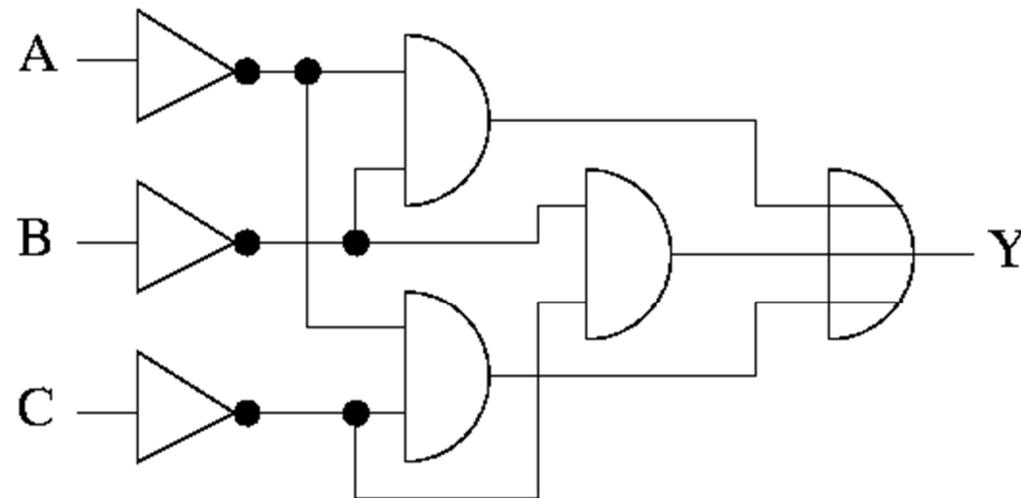
Beispiel einer logischer Schaltung

- 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:



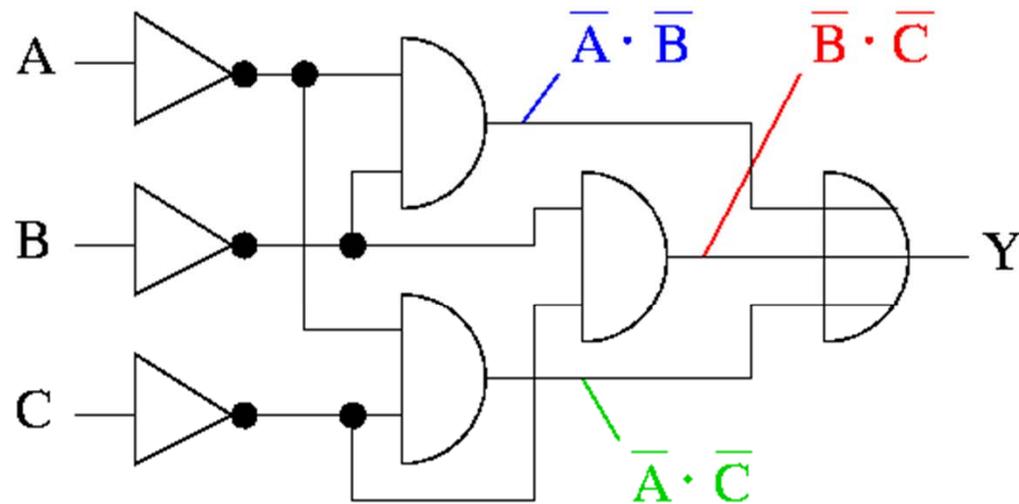
Beispiel einer logischen Schaltung

- 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:



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



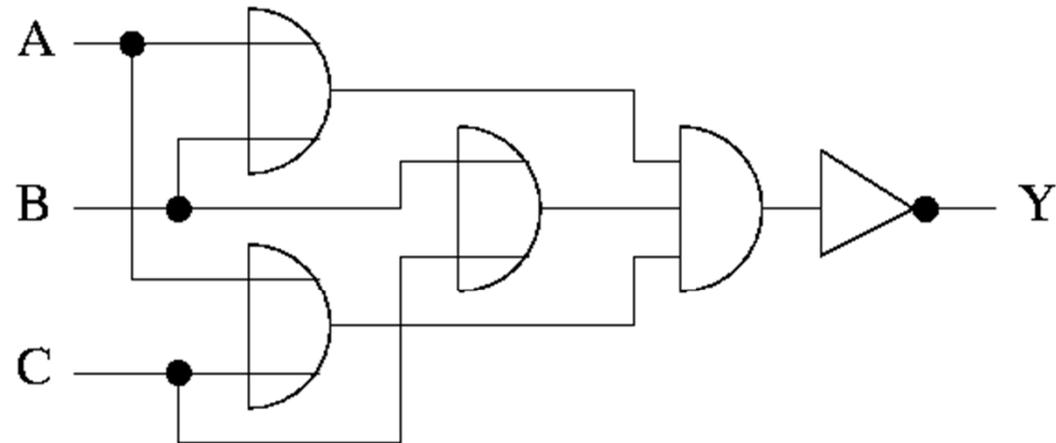
Beispiel einer logischer Schaltung

- 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:



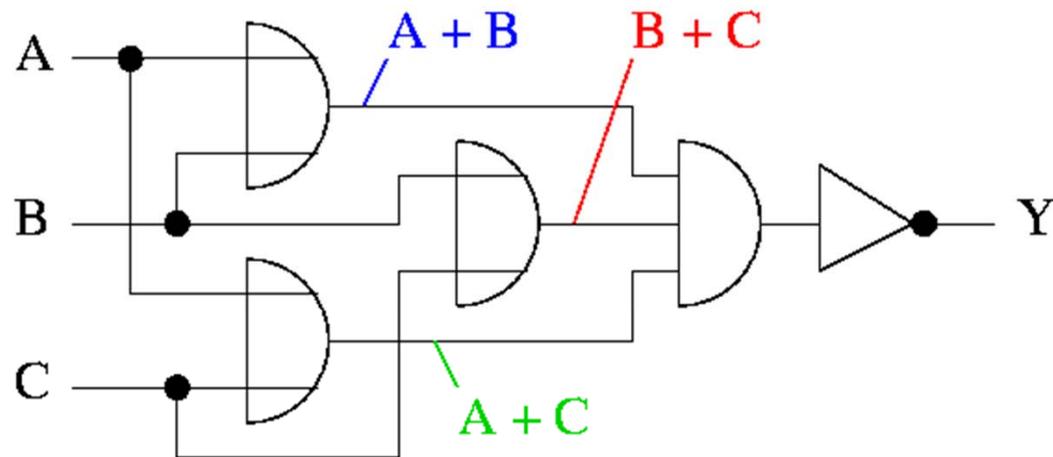
Beispiel einer logischer Schaltung

- 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:

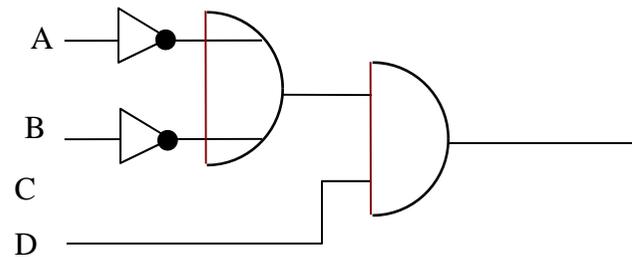
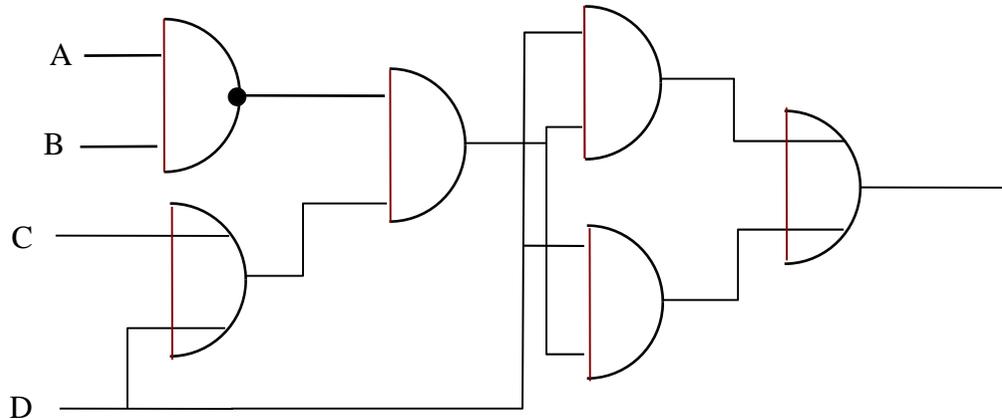


$$Y = \overline{(A + B) \cdot (B + C) \cdot (A + C)}$$

- 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, ...)



Synthese

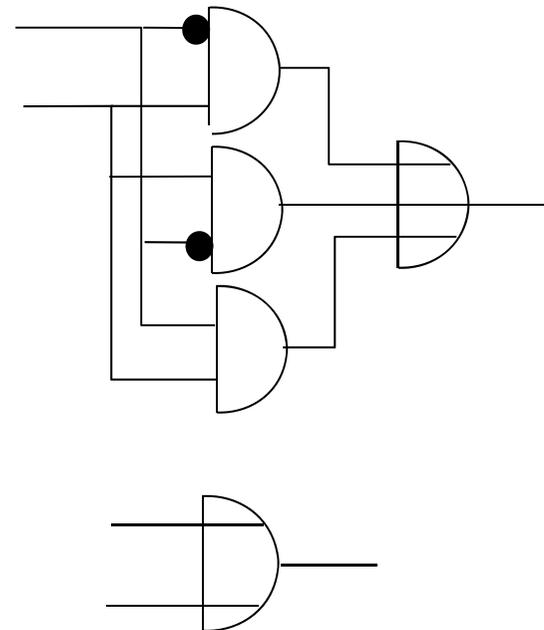
- eine **KDNF** ist günstiger als eine KKNF, wenn nur für wenige Kombinationen der Eingabewerte $f(x_1, x_2, \dots, x_n) = 1$ gilt. Umgekehrt, wenn nur für wenige Kombinationen der Eingabewerte $f(x_1, x_2, \dots, x_n) = 0$ gilt, ist die **KKNF** vorzuziehen.
- **Beispiel** : ODER-Funktion

Wahrheitstafel

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

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

$$\text{KKNF: } x + y$$



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?

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 böartig"
- 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 böseartig"
- V steht für: "Superman verhindert Böses"

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

$$(3) \neg W \rightarrow B$$

"Superman verhindert nicht das Böse"

$$(4) \neg V$$

"Wenn Superman existiert, dann ist er weder böseartig 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 böartig"
- V steht für: "Superman verhindert Böses"

Ist die folgende Formel erfüllbar?

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

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 $i, j, k \in \{1, 2, \dots, 9\}$.

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

Wertebelegungen beschreiben Beschriftungen des Gitters.

Ziel:

Für jede Anfangsbeschriftung eine Formelmengenge 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

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

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

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

Modell von F : \mapsto korrekte Lösung.

Frage: Ist F erfüllbar?

Überprüfung: Resolution oder Tableaux

Anwendungsbereiche

3. Planen

Ähnliches Problem: Planen

Beispiel: Modellierung eines Zeitplanungs-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 Zeitplanungs-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: $(P_{1,a,M,m} \vee P_{1,a,K,m}) \wedge (P_{1,b,S,d} \vee P_{1,b,K,d})$
 $(P_{2,a,S,d} \vee P_{2,a,K,d}) \wedge (P_{2,b,S,d} \vee P_{2,b,K,d})$
 $(P_{3,a,M,m} \vee P_{3,a,K,m}) \wedge (P_{3,b,S,d} \vee P_{3,b,K,d})$
 $\neg(P_{1,a,K,m} \wedge P_{1,b,K,d}) \wedge \neg(P_{2,a,K,d} \wedge P_{2,b,K,d}) \wedge \neg(P_{2,a,S,d} \wedge P_{2,b,S,d}) \wedge$
 $\neg(P_{3,a,K,m} \wedge P_{3,b,K,m}) \wedge (P_{1,a,M,m} \wedge P_{1,b,M,m}) \dots$

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:

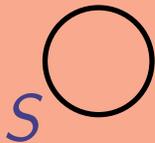
- Startzustand (direkt nach dem Einschalten).
- Benutzer gibt Telefonnummer ein.
- Telefonnummer ist vollständig eingegeben;
Handy versucht, Basisstation zu erreichen.
- Kontakt zur Basisstation aufgenommen;
anderer Teilnehmer wird angerufen.
- Gespräch wird geführt.

...

Beispiel 3: Handy

Wir stellen die Zustände durch Kreise dar.

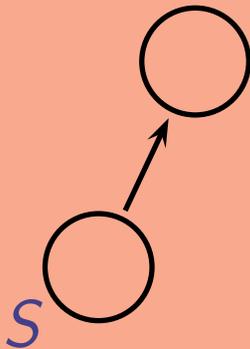
Wenn es möglich ist, von einem Zustand in einen anderen zu wechseln, zeichnen wir einen Pfeil dazwischen:



Beispiel 3: Handy

Wir stellen die Zustände durch Kreise dar.

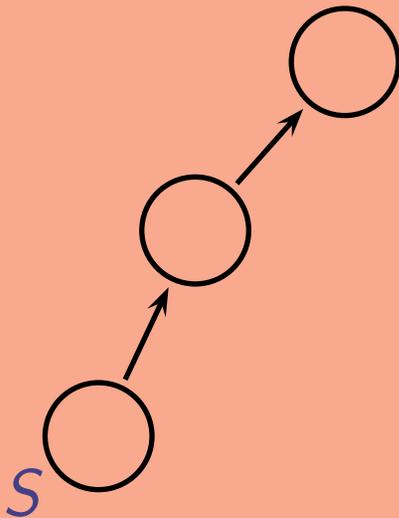
Wenn es möglich ist, von einem Zustand in einen anderen zu wechseln, zeichnen wir einen Pfeil dazwischen:



Beispiel 3: Handy

Wir stellen die Zustände durch Kreise dar.

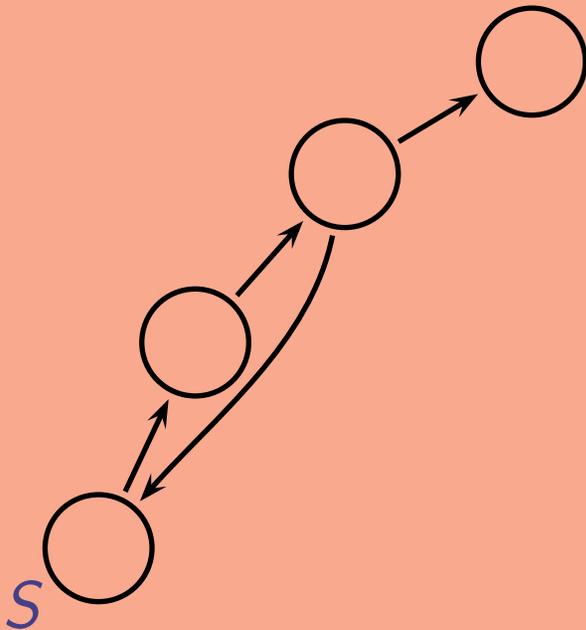
Wenn es möglich ist, von einem Zustand in einen anderen zu wechseln, zeichnen wir einen Pfeil dazwischen:



Beispiel 3: Handy

Wir stellen die Zustände durch Kreise dar.

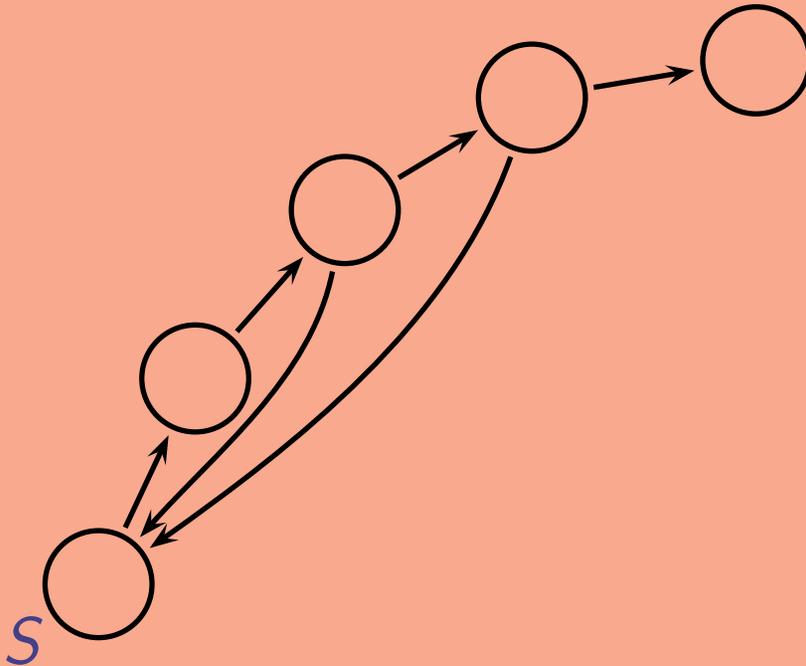
Wenn es möglich ist, von einem Zustand in einen anderen zu wechseln, zeichnen wir einen Pfeil dazwischen:



Beispiel 3: Handy

Wir stellen die Zustände durch Kreise dar.

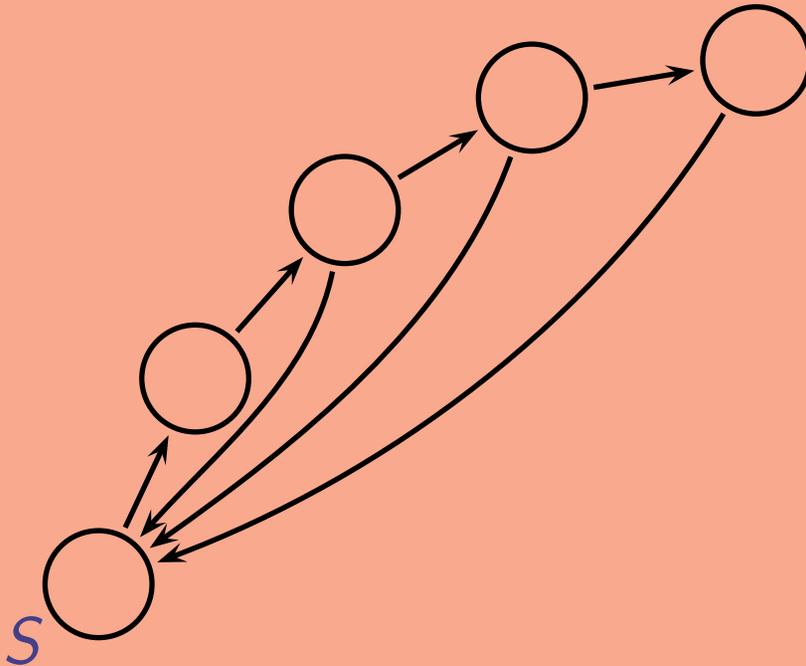
Wenn es möglich ist, von einem Zustand in einen anderen zu wechseln, zeichnen wir einen Pfeil dazwischen:



Beispiel 3: Handy

Wir stellen die Zustände durch Kreise dar.

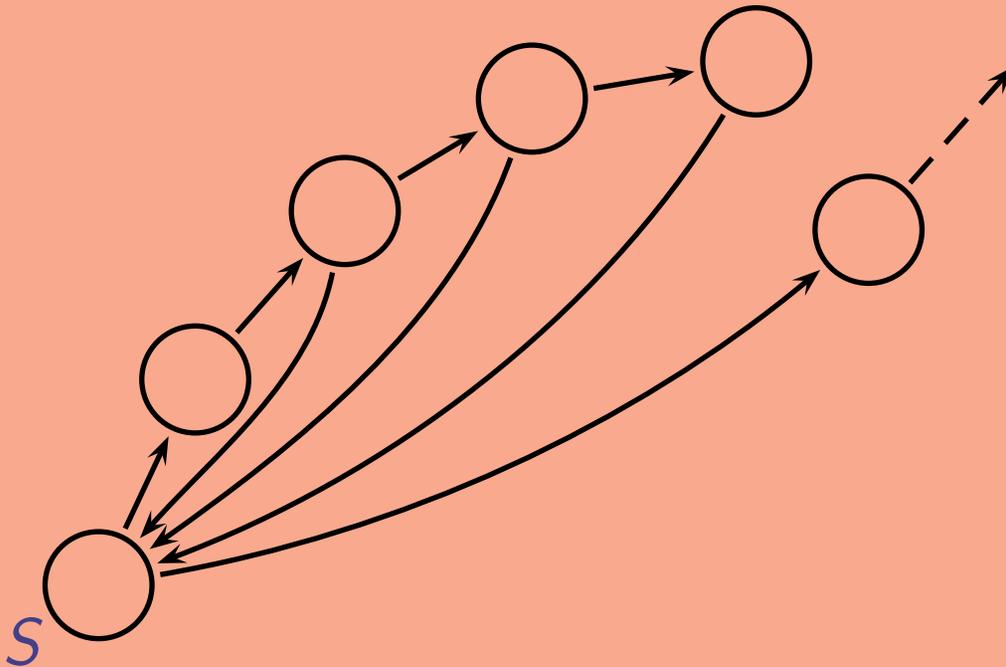
Wenn es möglich ist, von einem Zustand in einen anderen zu wechseln, zeichnen wir einen Pfeil dazwischen:



Beispiel 3: Handy

Wir stellen die Zustände durch Kreise dar.

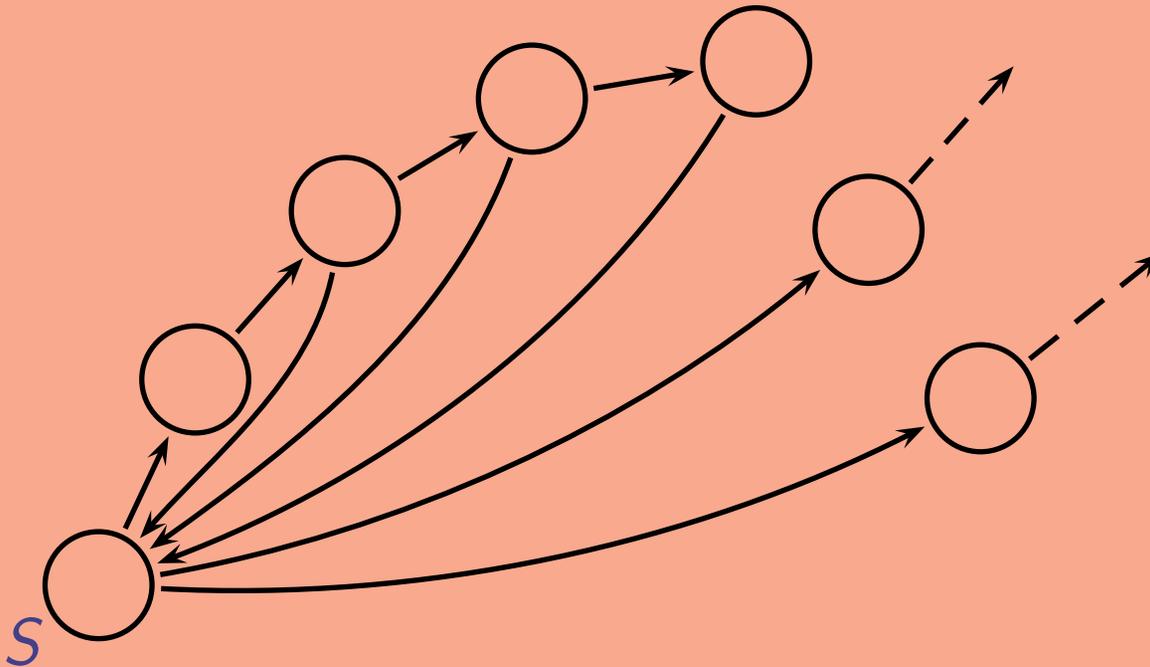
Wenn es möglich ist, von einem Zustand in einen anderen zu wechseln, zeichnen wir einen Pfeil dazwischen:



Beispiel 3: Handy

Wir stellen die Zustände durch Kreise dar.

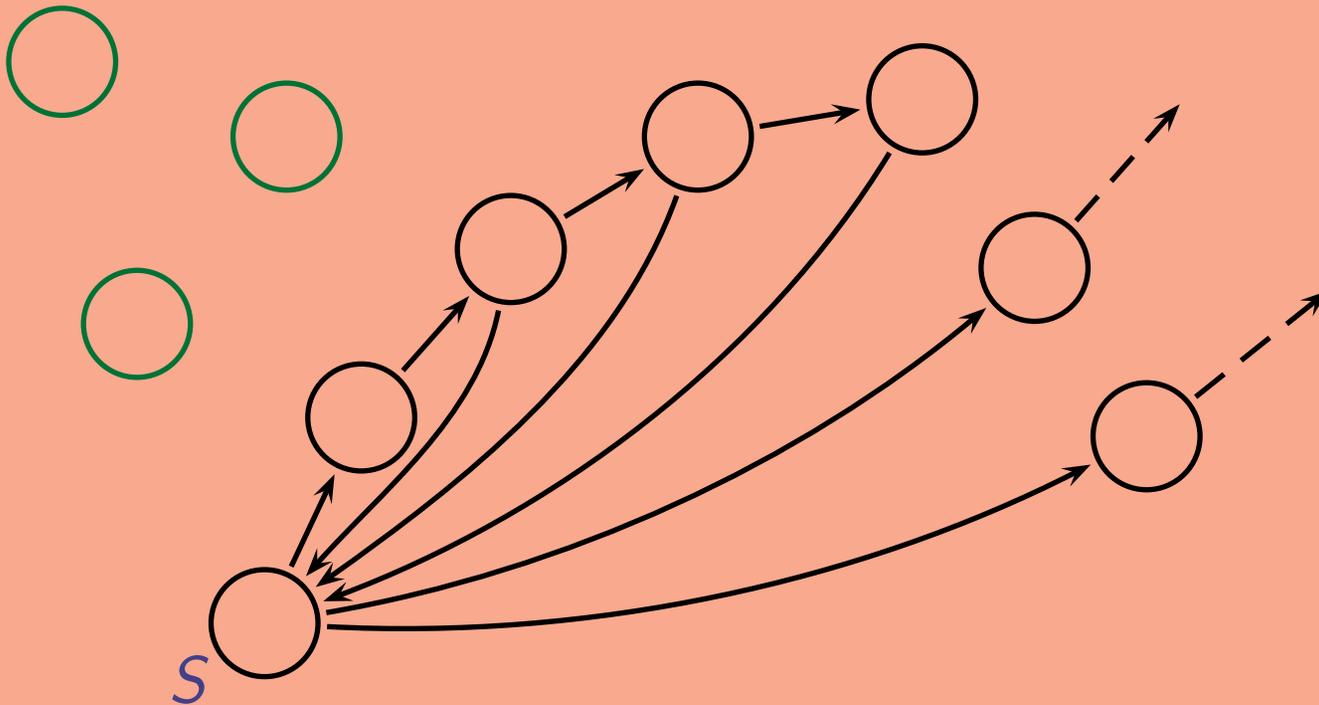
Wenn es möglich ist, von einem Zustand in einen anderen zu wechseln, zeichnen wir einen Pfeil dazwischen:



Beispiel 3: Handy

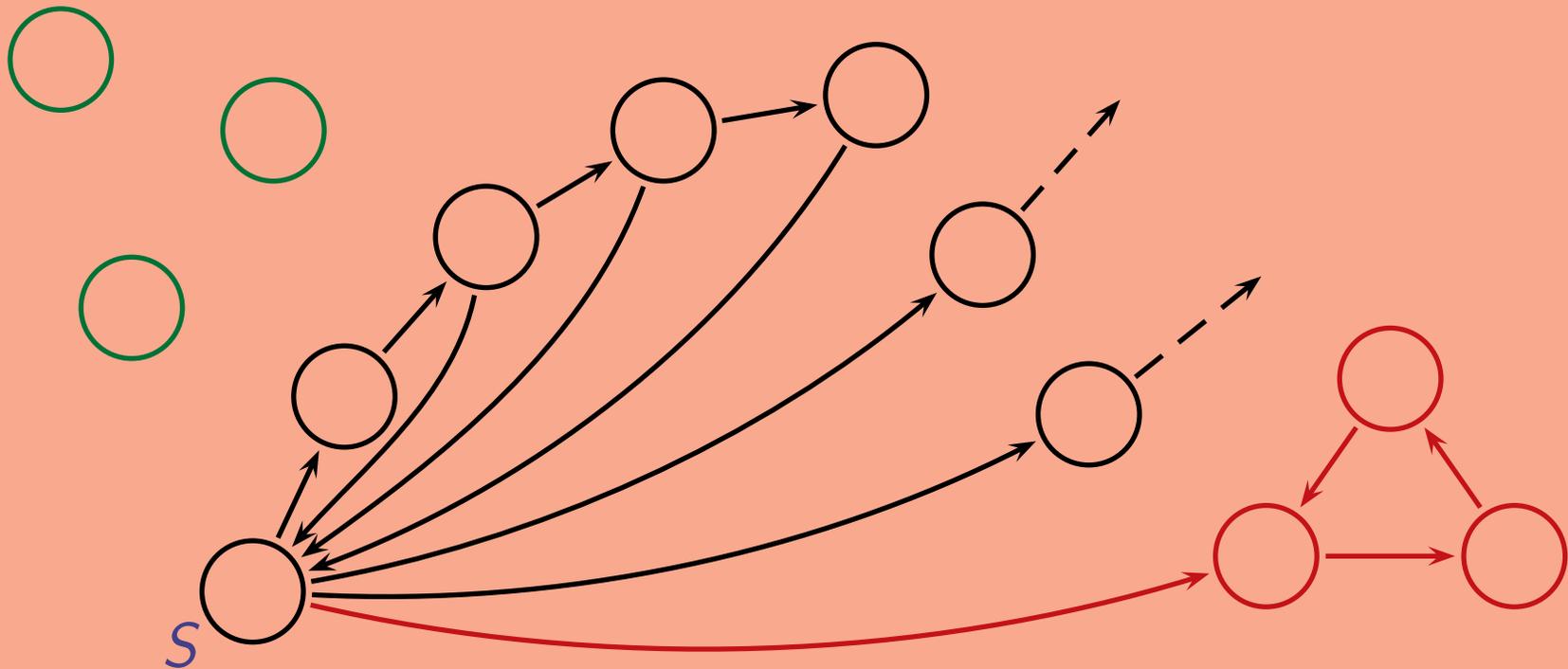
Es kann auch „unmögliche“ Zustände geben.

Aber diese dürfen dann natürlich nicht vom Startzustand aus erreichbar sein.



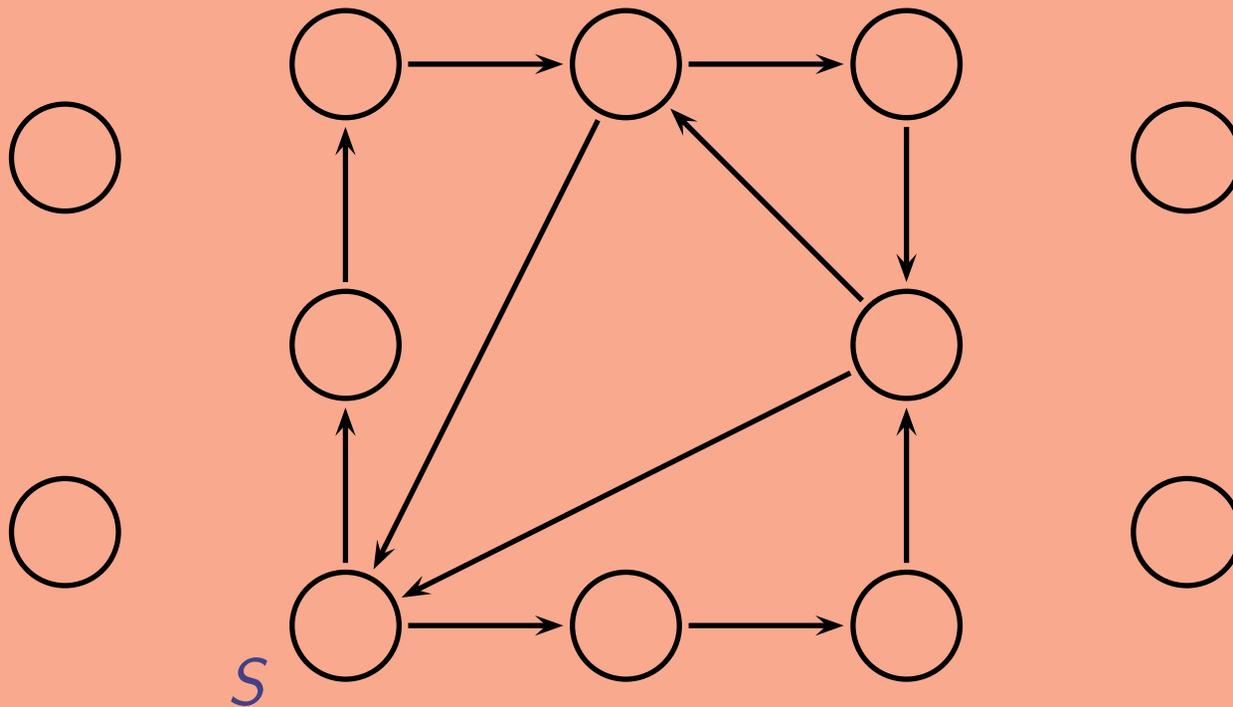
Beispiel 3: Handy

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



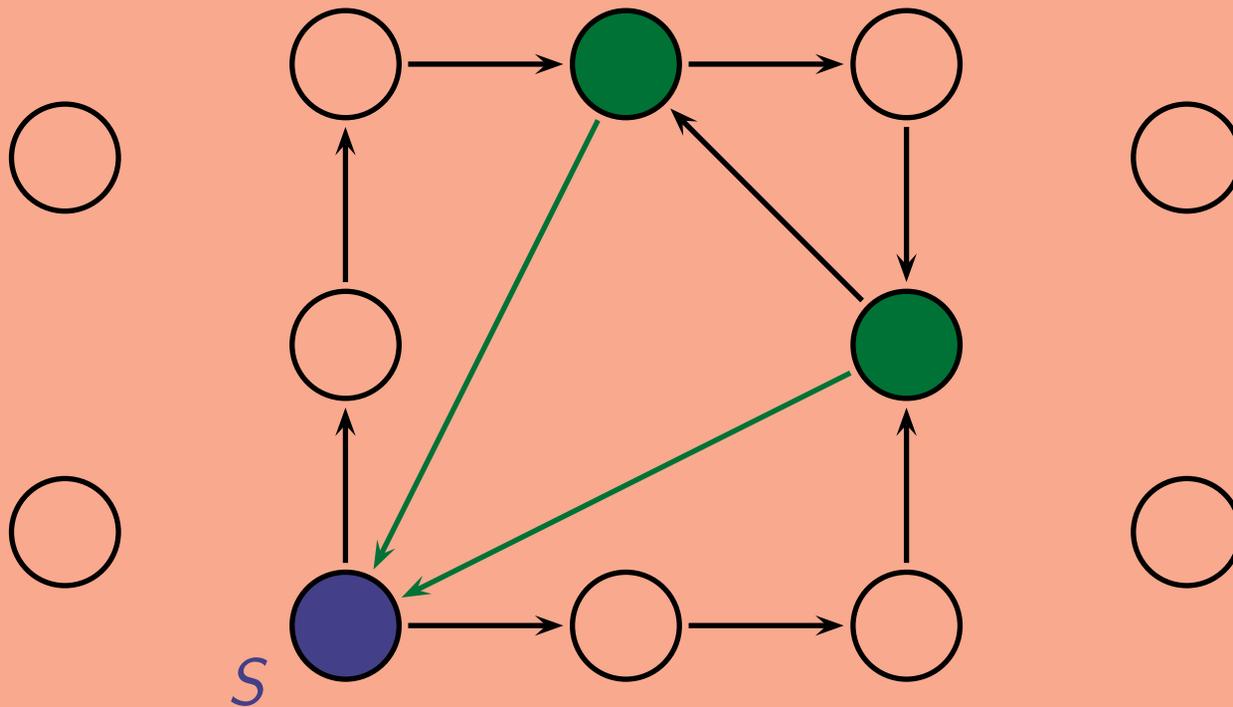
Beispiel 3: Handy

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



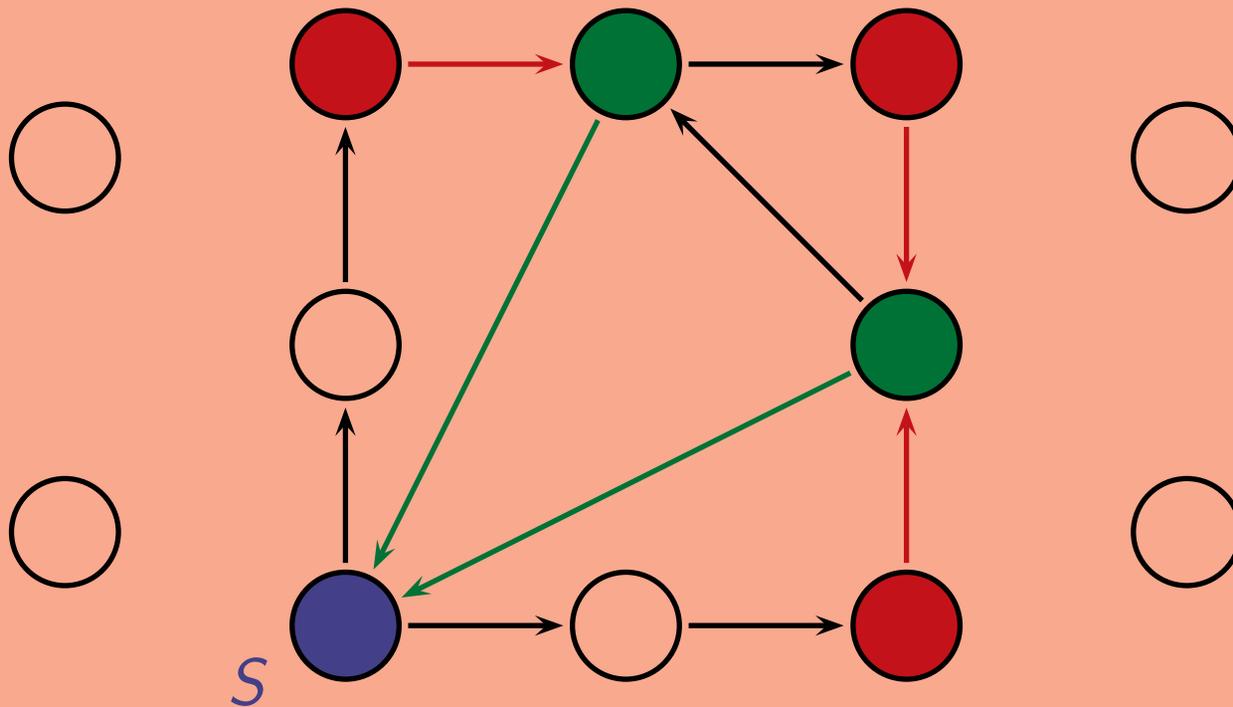
Beispiel 3: Handy

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



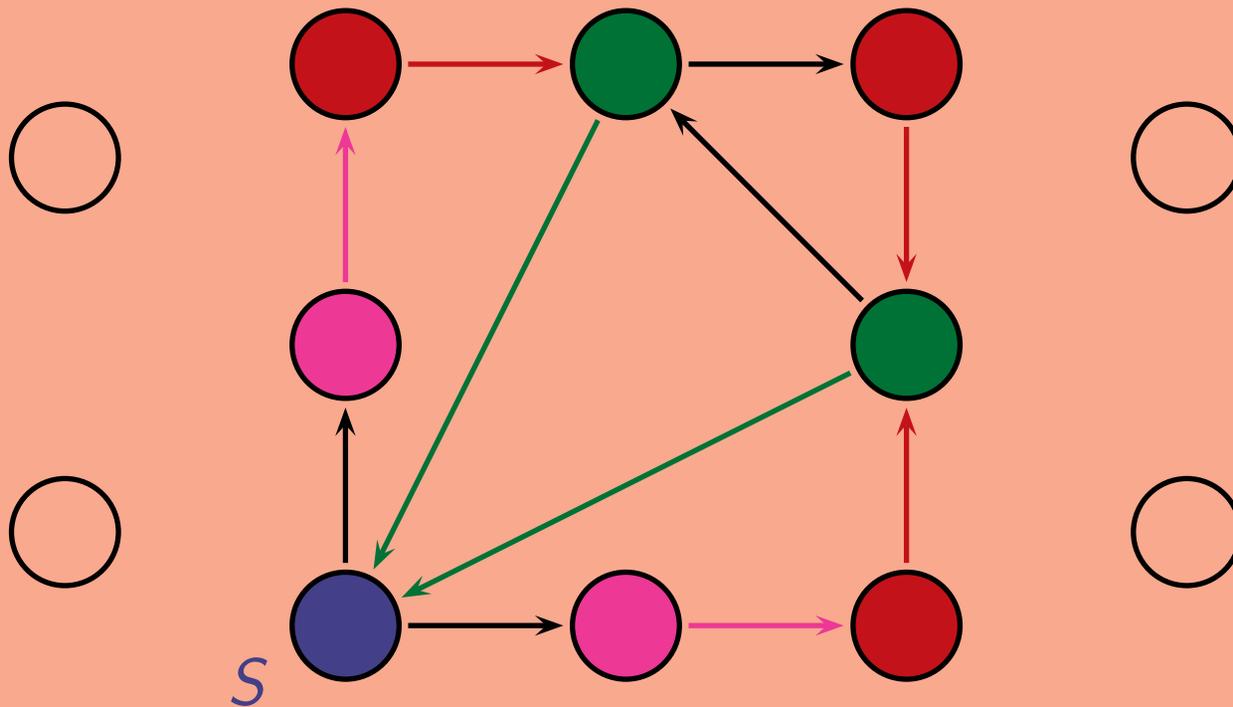
Beispiel 3: Handy

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



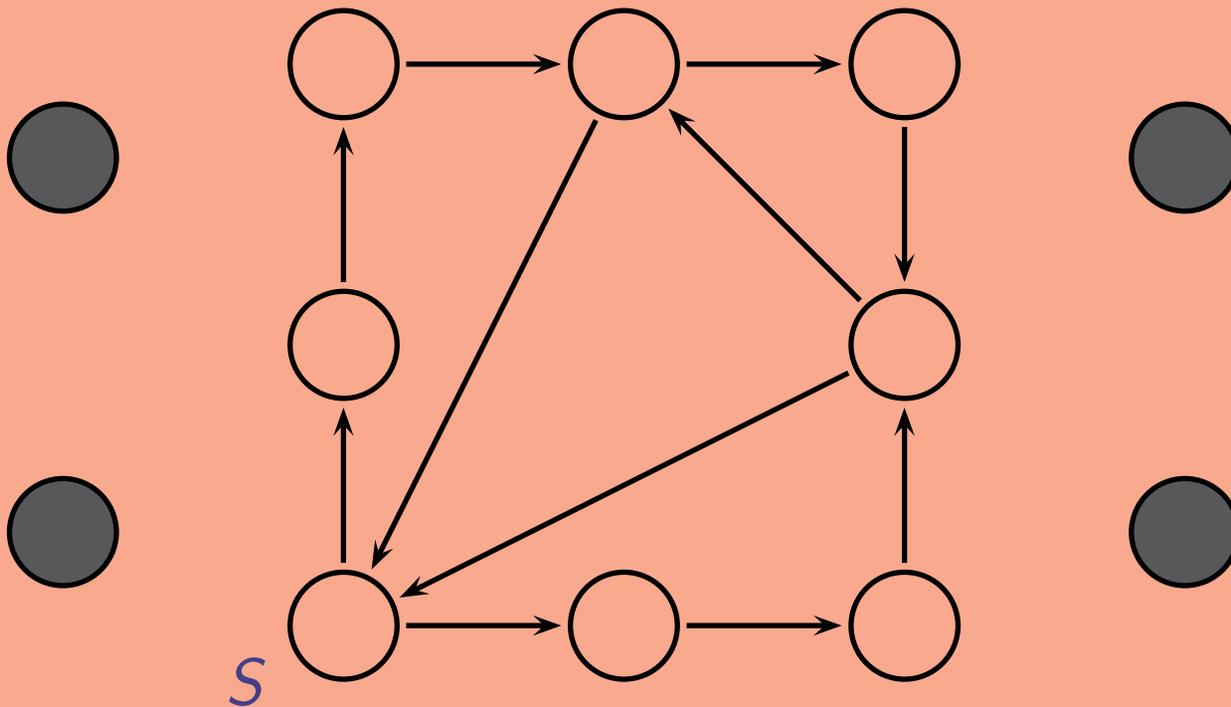
Beispiel 3: Handy

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



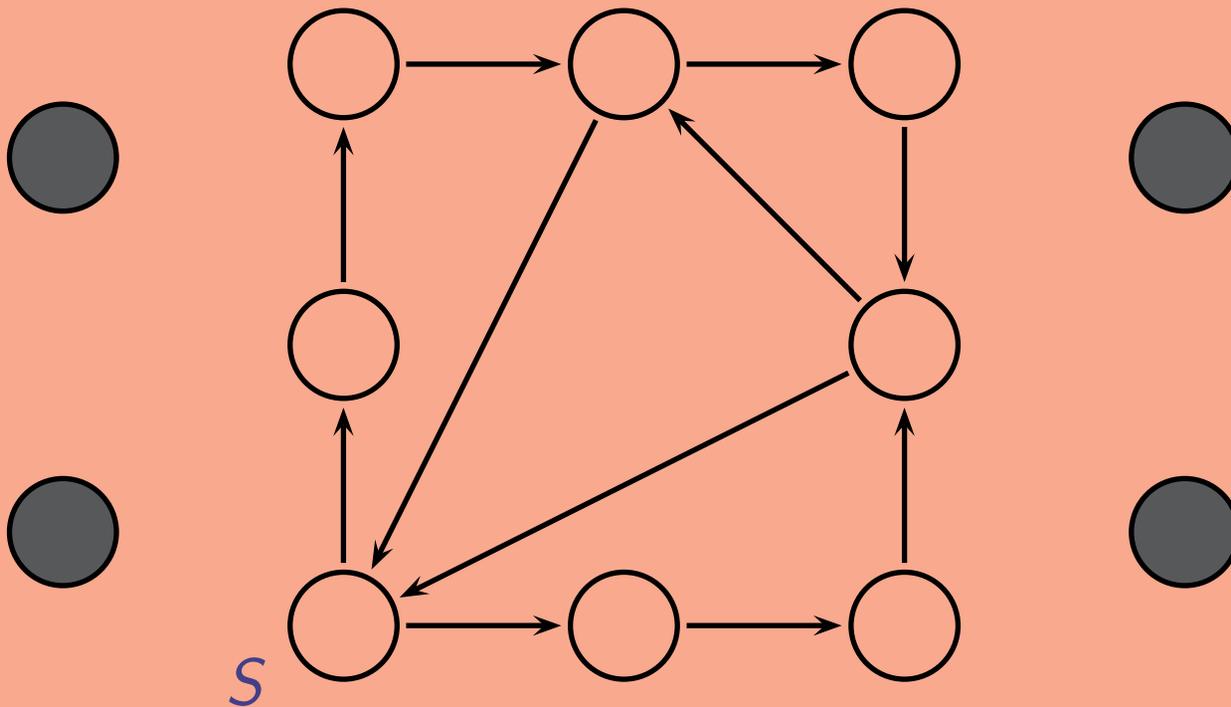
Beispiel 3: Handy

Zum Schluß ermitteln wir alle Zustände mit der Eigenschaft $EF \neg EF S$, 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 Übergänge 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

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

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

Softwareverifikation

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

Kodierung als Formeln

- (Mengen von) Elementen von S : Formeln

$s \in S \mapsto F_s := L_1 \wedge \dots \wedge L_n, \text{ wobei}$

$L_i = p_i$ 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)

$U = \{s_1, \dots, s_m\} \subseteq S \mapsto F_U = F_{s_1} \vee \dots \vee F_{s_m}$

Softwareverifikation

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

Kodierung als Formeln

- **Transitionsrelation** \rightarrow : Formel F_{\rightarrow}

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

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

Eine Transition von s nach s' :

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

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 ($F_{s'}(p'_1, \dots, p'_n)$)

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

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

Transitionen \mapsto neue Aussagenvariablen wie vorher.

(nicht sehr expressiv!)

Anwendungsbereiche

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