Aussagenlogik

Formale Logik

- Ziel: Formalisierung des Wahrheitsgehalts komplexer Aussagen
  - Welche Regeln erlauben korrekte Schlussfolgerungen
- Aussagenlogik
  - Klassisch: „Formeln“ bestehen nur aus Variablen, $\sim$, $\wedge$, $\vee$ und Klammern.
- Prädikatenlogik
  - 1. Stufe: Erweiterung der Aussagenlogik um Funktionssymbole, Prädikate und Quantoren über Variablen („es existiert“, „für alle“)
  - Höhere Stufen: weitere Quantoren, erweiterte Semantik
- V.g. Boole’schen Ausdrücke stellen eine bequeme Erweiterung klassischer aussagenlogischer Formeln dar, sind aber nicht ausdrucksstärker

Zweiwertigkeit, Verknüpfung von Aussagen

- Die klassische Formale Logik ist *zweiwertig*, beschäftigt sich also mit Aussagen, die entweder wahr oder falsch sind.
  - „Die Uhr geht richtig.“
  - „Ich habe Hunger.“
- Mehrere *atomare* Aussagen können zu neuen *komplexen* Aussagen verknüpft werden.
  - “Ich komme gerade aus der Vorlesung *und* ich bin total happy.”
  - Ob eine so entstandene Aussage wahr oder falsch ist, ergibt sich aus dem Wahrheitswert der ursprünglichen Aussagen und der verwendeten Verknüpfung.

Wahrheitswerte atomarer Aussagen

- Die Wahrheitswerte *wahr* und *falsch* von Aussagen lassen sich durch die Zahlen 1 bzw. 0 repräsentieren.
- **Definition**. Sei V eine endliche Menge atomarer Aussagen. Eine *Belegung* ist eine Abbildung $\beta: V \to \{0, 1\}$.
- Ein bequemer Weg, eine Belegung anzugeben, ist, die Werte der Aussagen $X_1, \dots, X_n \in \mathrm{V}$ in einem Vektor zusammenzufassen.
- **Definition.** Sei $a = (a_1, \dots, a_n)$ mit $a_i \in \{0, 1\}$ für alle $1 \leq i \leq n$. Die Belegung β_a ist definiert als $\beta(X_i) = a_i$ für alle $1 \leq i \leq n$.

Beispiele

- β könnte z.B. wie folgt definiert sein:

$$X_1 \mapsto 0,$$
$$X_2 \mapsto 0,$$
$$X_3 \mapsto 1,$$
$$X_4 \mapsto 0,$$
$$X_5 \mapsto 1,$$
$$X_6 \mapsto 1,$$
$$X_7 \mapsto 1,$$
$$\dots$$

- Sei $a = (1,0,0,1,1,0,1,0, \dots)$
  Dann ist β_a:

$$X_1 \mapsto 1,$$
$$X_2 \mapsto 0,$$
$$X_3 \mapsto 0,$$
$$X_4 \mapsto 1,$$
$$X_5 \mapsto 1,$$
$$X_6 \mapsto 0,$$
$$X_7 \mapsto 1,$$
$$X_8 \mapsto 0,$$
$$\dots$$

Wahrheitswerte komplexer Aussagen

- Um die Wahrheitswerte weiterer, aus atomaren Aussagen $X_1, X_2, \ldots$ zusammengesetzter Aussagen (z.B. “sowohl X_1 als auch X_2”) bestimmen zu können, benötigen wir:
  1. eine genaue Definition, wie komplexe Aussagen aus elementaren Aussagen gebildet werden können (Syntax)
  2. einen Mechanismus, der den Wahrheitswert (Semantik) so gebildeter Aussagen aus den Wahrheitswerten der enthaltenen Aussagen herleitet
- Für Punkt 1. verwenden wir vollständig geklammerte Boole’sche Ausdrücke.
  - Bei diesen soll die Menge $V = \{X_1, \ldots, X_n\}$ dann ebenfalls atomare Aussagen repräsentieren.
- Für Punkt 2. führen wir zunächst ein Hilfskonstrukt ein.

Schaltfunktionen

- **Definition.** Sei $n \in \mathbb{N}$. Ein *n-stellige Schaltfunktion* ist eine Abbildung $f: \{0,1\}^n \to \{0,1\}$.
- Beispiele.
  - Die Funktion $\sim: \{0,1\} \to \{0,1\}, x \mapsto 1 - x$ heißt *Negation*.
  - Die zwei binären Schaltfunktionen $\wedge$ (*Konjunktion*) und $\vee$ (*Disjunktion*) sind durch folgende Tabelle definiert:

    (Wir schreiben $\wedge$ und $\vee$ typischerweise in Infix-Notation.)

$\boldsymbol{x} \in \{\mathbf{0},\mathbf{1}\}^{\mathbf{2}}$	$\mapsto$	$\wedge(\boldsymbol{x}) \in \{\mathbf{0},\mathbf{1}\}$	$\vee(\boldsymbol{x}) \in \{\mathbf{0},\mathbf{1}\}$
$(0,0)$		0	0
$(0,1)$		0	1
$(1,0)$		0	1
$(1,1)$		1	1

Anmerkung: Überladung von Symbolen

- Wir haben die Symbole $\sim$, $\wedge$, $\vee$, f nun in zwei unterschiedlichen Kontexten mit unterschiedlicher Bedeutung verwendet:
  1. Als Zeichen in vollständig geklammerten Boole'schen Ausdrücken
  2. Als Namen bestimmter Schaltfunktionen
- Mehrfachverwendungen sind aber nichts Ungewöhnliches, siehe z.B.:
  - h → Abk. für „Stunde"; SI-Präfix für 100; Planckssches Wirkungsquantum
  - V → Formelzeichen für „Volumen"; SI-Einheit für „Volt"
- Wir müssen nur aufpassen, dass wir uns immer über den richtigen Kontext im Klaren sind.

Werte von v.g. Boole'schen Ausdrücken

- **Definition.** Die Abbildung $\phi_\beta: B \to \{0,1\}$ heißt *Wert bzgl. der Belegung* β und ist wie folgt rekursiv definiert:

$\boldsymbol{x}$	$\boldsymbol{\phi_\beta(x)}$
$\mathbf{0}$	0
$\mathbf{1}$	1
$X_i \in V$	$\beta(X_i)$
$(\sim b)$	$\sim\left(\phi_\beta(b)\right)$
$(b_1 \wedge b_2)$	$\phi_\beta(b_1) \wedge \phi_\beta(b_2)$
$(b_1 \vee b_2)$	$\phi_\beta(b_1) \vee \phi_\beta(b_2)$
$f(b_1, \ldots, b_k)$ mit $f \in \hat{F}$	$f(\phi(b_1), \ldots, \phi(b_k))$, falls f eine k-stellige Schaltfunktion bezeichnet

Quiz: welche $\sim$, $\wedge$, $\vee$, f gehören zu v.g. Boole'schen Ausdrücken, welche bezeichnen Schaltfunktionen?

Beispiele

- Sei β eine Belegung, für die gilt: $\beta(X_1) = 0$ und $\beta(X_2) = 1$.
- Dann ist
  - $\phi_\beta\big((X_1 \wedge X_2)\big) = \phi_\beta(X_1) \wedge \phi_\beta(X_2) = \beta(X_1) \wedge \beta(X_2) = 0 \wedge 1 = 0$
- Sei außerdem $g\colon \{0,1\}^2 \to \{0,1\}$ eine wie folgt definierte Schaltfunktion: $(0,0) \mapsto 0, (0,1) \mapsto 1, (1,0) \mapsto 0, (1,1) \mapsto 1$. Dann folgt:
  - $\phi_\beta\left(\left(0 \vee \left(\sim g\big(X_2, (\sim X_1)\big)\right)\right)\right)$

$$
\begin{aligned}
&= \phi_\beta(0) \vee \phi_\beta\left(\left(\sim g\big(X_2, (\sim X_1)\big)\right)\right) && = 0 \vee \sim\left(g\big(1, \sim(0)\big)\right)\\
&= 0 \vee \sim\left(\phi_\beta\left(g\big(X_2, (\sim X_1)\big)\right)\right) && = 0 \vee \sim\big(g(1,1)\big)\\
&= 0 \vee \sim\left(g\left(\phi_\beta(X_2), \phi_\beta\big((\sim X_1)\big)\right)\right) && = 0 \vee \sim(1)\\
&= 0 \vee \sim\left(g\left(\beta(X_2), \sim\left(\phi_\beta(X_1)\right)\right)\right) && = 0 \vee 0\\
&= 0 \vee \sim\left(g\left(1, \sim\big(\beta(X_1)\big)\right)\right) && = 0
\end{aligned}
$$

Quiz

- Für welche Belegungen β gilt: $\phi_\beta\left(\left(X_1 \wedge (\sim X_2)\right)\right) = 1$?
- $$\phi_\beta\left(\left(X_1 \wedge (\sim X_2)\right)\right) = \phi_\beta(X_1) \wedge \phi_\beta\left((\sim X_2)\right)$$
$$= \beta(X_1) \wedge \sim\left(\phi_\beta(X_2)\right)$$
$$= \beta(X_1) \wedge \sim\left(\beta(X_2)\right)$$
- Der Wert von $\wedge\,(b_1, b_2)$ ist nur dann 1, wenn $b_1 = b_2 = 1$ gilt.

  $\Rightarrow \beta(X_1) = 1$ und $\sim\left(\beta(X_2)\right) = 1 \Leftrightarrow \beta(X_2) = 0$

Durch v.g.B.A. berechnete Funktion

- Für unterschiedliche Belegungen β kann der durch ϕ_β berechnete Wert eines v.g. Boole'schen Ausdrucks unterschiedlich sein.
- Abkürzung: Ist die Belegung β mittels eines Vektors a definiert, schreiben wir statt ϕ_{β_a} kürzer ϕ_a.
- **Definition.** Sei b ein vollständig geklammerter Boole'scher Ausdruck. Die Schaltfunktion $f_b: \{0,1\}^n \to \{0,1\},\ x \mapsto \phi_x(b)$ heißt *die durch den Ausdruck b berechnete Funktion.*

Beispiel

- Sei $b = \big((\sim X_1) \wedge (\sim X_2)\big)$

a_1	a_2	$f_b(a_1, a_2)$
0	0	1
0	1	0
1	0	0
1	1	0

Äquivalenz

- Manche vollständig geklammerte Boole'sche Ausdrücke wie z.B. $(X_1 \wedge X_2)$ und $(X_2 \wedge X_1)$ sind als Worte der Sprache B zwar nicht identisch, berechnen aber dieselbe Funktion.
  - Das kennen wir von natürlichen Sprachen, z.B.:
    - „Als er den Inhalt sah, lächelte er selig: Süßigkeiten liebte Herr Meyer."
    - „Als er den Inhalt sah, lächelte er selig: Herr Meyer liebte Süßigkeiten."
- **Definition** (Äquivalenz v.g. Boole'scher Ausdrücke). Die Relation $\equiv \subseteq B \times B$ ist wie folgt definiert. Seien b_1, b_2 zwei v.g. Boole'sche Ausdrücke; es gilt $b_1 \equiv b_2$ genau dann, wenn $f_{b_1} = f_{b_2}$.

Beispiel

- Wir zeigen, dass gilt: $(X_1 \wedge X_2) \equiv (X_2 \wedge X_1)$.
- Damit dies der Fall ist, muss gelten: $f_{(X_1 \wedge X_2)} = f_{(X_2 \wedge X_1)}$
- Sei $a \in \{0,1\}^n$ beliebig. Dann gilt:

$$\begin{aligned} f_{(X_1 \wedge X_2)}(a) &= \phi_a\big((X_1 \wedge X_2)\big) \\ &= \phi_a(X_1) \wedge \phi_a(X_2) \\ &= \phi_a(X_2) \wedge \phi_a(X_1) \\ &= \phi_a\big((X_2 \wedge X_1)\big) \\ &= f_{(X_2 \wedge X_1)}(a) \end{aligned}$$

qed.

Nützliche Äquivalenzen

- $(b \wedge b) \equiv b$
- $(b \vee b) \equiv b$
- $(b_1 \wedge b_2) \equiv (b_2 \wedge b_1)$
- $(b_1 \vee b_2) \equiv (b_2 \vee b_1)$
- $\big((b_1 \wedge b_2) \wedge b_3\big) \equiv \big(b_1 \wedge (b_2 \wedge b_3)\big)$
- $\big((b_1 \vee b_2) \vee b_3\big) \equiv \big(b_1 \vee (b_2 \vee b_3)\big)$
- $\big(b_1 \wedge (b_1 \vee b_2)\big) \equiv b_1$
- $\big(b_1 \vee (b_1 \wedge b_2)\big) \equiv b_1$
- $\big(b_1 \wedge (b_2 \vee b_3)\big) \equiv \big((b_1 \wedge b_2) \vee (b_1 \wedge b_3)\big)$
- $\big(b_1 \vee (b_2 \wedge b_3)\big) \equiv \big((b_1 \vee b_2) \wedge (b_1 \vee b_3)\big)$
- $\big(\sim(\sim b)\big) \equiv b$
- $\big(\sim(b_1 \wedge b_2)\big) \equiv \big((\sim b_1) \vee (\sim b_2)\big)$
- $\big(\sim(b_1 \vee b_2)\big) \equiv \big((\sim b_1) \wedge (\sim b_2)\big)$

} Regeln von DeMorgan

Boole’sche Ausdrücke ohne Klammern

- V.g.B.A. wären lesbarer, wenn es nicht so viele Klammern gäbe, aber einfach weglassen ist problematisch:

$$X_1 \vee X_2 \wedge X_3$$

$$((X_1 \vee X_2) \wedge X_3) \qquad (X_1 \vee (X_2 \wedge X_3))$$

- Als Lösung verabreden wir folgende Operatorpräzedenzen:
  - $\sim$ bindet stärker als $\wedge$
  - $\wedge$ bindet stärker als $\vee$
- Bei einem Ausdruck wie z.B. $X_1 \wedge X_2 \wedge X_3$ ist dann aber immer noch nicht klar, ob $\big((X_1 \wedge X_2) \wedge X_3\big)$ oder $\big(X_1 \wedge (X_2 \wedge X_3)\big)$ gemeint ist. Da solche Ausdrücke aber äquivalent sind, ist das nicht schlimm.

Beispiele von Boole’schen Ausdrücken ohne Klammern

- $X_1 \wedge X_2$
- $(X_2 \vee \mathbf{0}) \wedge \sim X_3$
- $X_2 \vee \mathbf{0} \wedge \sim X_3$

- Aus offensichtlichen Gründen nennen wir diese abgekürzten Boole’schen Ausdrücke dann nicht mehr „vollständig geklammert“.

Modell eines Boole’schen Ausdrucks

- **Definition.** Sei b ein v.g. Boole’scher Ausdruck und sei β eine Belegung. Falls gilt $\phi_\beta(b) = 1$, dann heißt β ein *Modell für* b, und wir schreiben auch $\beta \vDash b$. Gilt hingegen $\phi_\beta(B) = 0$, dann ist β kein Modell für b, und wir schreiben $\beta \nvDash b$.
  - Ist die Belegung β ein Modell für einen Boole’schen Ausdruck b, so sagen wir auch: „b ist gültig unter β“.
- **Definition.** Ein v.g. Boole’scher Ausdruck b heißt *erfüllbar*, wenn es ein Modell für b gibt, ansonsten *unerfüllbar*.
- **Definition.** Ein v.g. Boole’scher Ausdruck b heißt *gültig* oder *Tautologie*, wenn jede Belegung ein Modell für b ist. Dann schreiben wir: $\vDash b$.

Beispiele

- Betrachten wir den B.A. $b_1 = X_1 \wedge \sim X_2$. Ist dieser erfüllbar?
  - Ja, denn eine Belegung β mit $\beta(X_1) = 1$ und $\beta(X_2) = 0$ ist ein Modell für b_1.
- Betrachten wir den B.A. $b_2 = X_1 \wedge \sim X_1$. Ist dieser erfüllbar?
  - Dazu müsste eine Belegung existieren mit $\beta(X_1) = 1$ und $\beta(X_1) = 0$.
  - Da es eine solche Belegung nicht gibt, ist b_2 also unerfüllbar.
- Betrachten wir den B.A. $b_3 = X_1 \vee \sim X_1$. Ist dieser erfüllbar?
  - Ja, und da $\phi_\beta(b_3) = 1$ sowohl für Belegungen mit $\beta(X_1) = 0$ also auch für solche mit $\beta(X_1) = 1$ gilt, ist b_3 sogar eine Tautologie.

Erfüllbarkeit, Gültigkeit, Negation

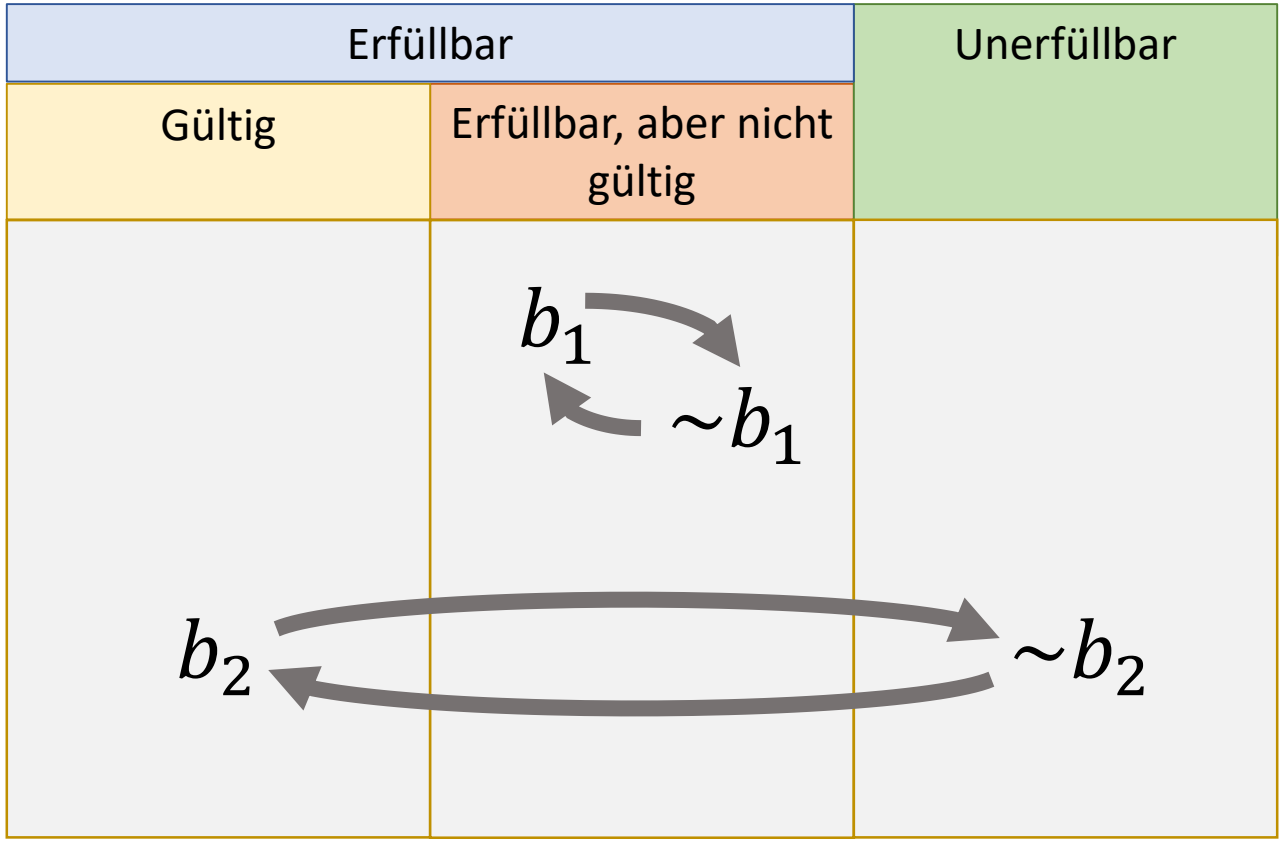
- **Satz.** Ein v.g. Boole'scher Ausdruck b ist genau dann gültig, wenn $(\sim b)$ unerfüllbar ist.
- Beweis.

$$b \text{ ist gültig}$$
$$\Leftrightarrow \text{jede Belegung } \beta \text{ ist ein Modell für } b$$
$$\Leftrightarrow \text{für jede Belegung } \beta \text{ gilt: } \phi_\beta(b) = 1$$
$$\Leftrightarrow \text{für jede Belegung } \beta \text{ gilt: } \phi_\beta\Big(\big(\sim(\sim b)\big)\Big) = 1$$
$$\Leftrightarrow \text{für jede Belegung } \beta \text{ gilt: } \phi_\beta\big((\sim b)\big) = 0$$
$$\Leftrightarrow \text{jede Belegung } \beta \text{ ist kein Modell für } (\sim b)$$
$$\Leftrightarrow (\sim b) \text{ ist unerfüllbar}$$

qed.

- **Korrolar.** Ein v.g. Boole'scher Ausdruck b ist erfüllbar, aber nicht gültig, genau dann, wenn $(\sim b)$ ebenfalls erfüllbar, aber nicht gültig ist.

Veranschaulichung Negation



Weitere notationelle Vereinfachungen

- Durch die Operatorpräzedenzen haben wir uns bereits einige Schreibarbeit gespart, aber es geht noch kompakter:
  - Statt $\sim X_i$ schreiben wir $\overline{X_i}$.
  - Einen Boole'schen Ausdruck der Form X_i oder $\overline{X_i}$ nennen wir auch *Literal*.
  - Statt $X_{i_1}^* \wedge X_{i_2}^* \wedge \cdots \wedge X_{i_k}^*$ schreiben wir kurz $X_{i_1}^* X_{i_2}^* \ldots X_{i_k}^*$, wenn alle $X_{i_j}^*$ Literale sind $(1 \leq j \leq k)$
- Damit vereinfacht sich zum Beispiel der Ausdruck

$$X_3 \wedge \sim X_{24} \vee \sim X_{13} \wedge \sim X_{15} \wedge X_{20}$$

zu

$$X_3 \overline{X_{24}} \vee \overline{X_{13} X_{15}} X_{20}$$

Weitere notationelle Vereinfachungen (2)

- **Definition**. Seien $b_j, \ldots, b_k$ Boole'sche Ausdrücke. Wir definieren:

$$\bigwedge_{i=j}^{k} b_i = b_j \wedge \cdots \wedge b_k$$

$$\bigvee_{i=j}^{k} b_i = b_j \vee \cdots \vee b_k$$

Eine Beobachtung bzgl. des Wahrheitswerts bei notationellen Vereinfachungen

- **Lemma.** Sei β eine beliebige Belegung. Dann gilt:
  - $\phi_\beta\left(\bigwedge_{i=1}^{k} b_i\right) = 1$ genau dann, wenn $\phi_\beta(b_i) = 1$ für alle $1 \leq i \leq k$ gilt.
  - $\phi_\beta\left(\bigvee_{i=1}^{k} b_i\right) = 0$ genau dann, wenn $\phi_\beta(b_i) = 0$ für alle $1 \leq i \leq k$ gilt.
- Den Beweis führt man durch eine simple Induktion über k.
- Offensichtlich gilt dann auch das Folgende:
- **Korollar.** Sei β eine beliebige Belegung. Dann gilt:
  - $\phi_\beta\left(\bigwedge_{i=1}^{k} b_i\right) = 0$ genau dann, wenn $\phi_\beta(b_i) = 0$ für mindestens ein i gilt, $1 \leq i \leq k$.
  - $\phi_\beta\left(\bigvee_{i=1}^{k} b_i\right) = 1$ genau dann, wenn $\phi_\beta(b_i) = 1$ für mindestens ein i gilt, $1 \leq i \leq k$.

Allgemeine Distributivgesetze

- Mit dieser Notation kann man wie im folgenden Satz Verallgemeinerungen der bekannten Äquivalenzen elegant formulieren.
- **Satz.** Seien $b_1, \dots b_k, c_1, \dots c_l$ Boole'sche Ausdrücke. Dann gilt:

$$1.\ \left(\bigwedge_{i=1}^{k} b_i\right) \vee \left(\bigwedge_{j=1}^{l} c_j\right) \equiv \bigwedge_{i=1}^{k}\left(\bigwedge_{j=1}^{l}\left(b_i \vee c_j\right)\right)$$

$$2.\ \left(\bigvee_{i=1}^{k} b_i\right) \wedge \left(\bigvee_{j=1}^{l} c_l\right) \equiv \bigvee_{i=1}^{k}\left(\bigvee_{j=1}^{l}\left(b_i \wedge c_j\right)\right)$$

Beweis.

Zu 1.:

$$\left(\bigwedge_{i=1}^{k} b_i\right) \vee \left(\bigwedge_{j=1}^{l} c_j\right) \equiv \mathbf{1}$$

$\Leftrightarrow$ $\left(\bigwedge_{i=1}^{k} b_i\right) \equiv \mathbf{1}$ oder $\left(\bigwedge_{j=1}^{l} c_j\right) \equiv \mathbf{1}$

$\Leftrightarrow$ $\forall b_i \in \{b_1, \dots, b_k\}: b_i \equiv \mathbf{1}$ oder $\forall c_j \in \{c_1, \dots, c_l\}: \mathrm{c_j} \equiv \mathbf{1}$

$\Leftrightarrow$ $\forall b_i \in \{b_1, \dots, b_k\}, c_j \in \{c_1, \dots, c_l\}: \left(b_i \vee c_j\right) \equiv \mathbf{1}$

$\Leftrightarrow$ $\bigwedge_{i=1}^{k}\left(\bigwedge_{j=1}^{l}\left(b_i \vee c_j\right)\right) \equiv \mathbf{1}$ ■

Zu 2.: analog

qed

Teilausdrücke

- **Definition.** Ein v.g.B.A. b' ist *Teilausdruck* eines v.g.B.A. b genau dann, wenn eine der folgenden Bedingungen zutrifft:
  - $b' = b$
  - $b = (\sim b_1)$ und b' ist Teilausdruck von b_1
  - $b = (b_1 \wedge b_2)$ und b' ist Teilausdruck von b_1 oder von b_2
  - $b = (b_1 \vee b_2)$ und b' ist Teilausdruck von b_1 oder von b_2
  - $b = f(b_1, \ldots, b_k)$ und b' ist Teilausdruck einer der v.g.B.A. $b_1, \ldots b_k$

Beispiel

- $\Big((\sim X_1) \vee \big(\mathbf{0} \wedge f(\mathbf{1})\big)\Big)$ enthält folgende Teilausdrücke:
  - $(\sim X_1)$
  - X_1
  - $\big(\mathbf{0} \wedge f(\mathbf{1})\big)$
  - $\mathbf{0}$
  - $f(\mathbf{1})$
  - $\mathbf{1}$

Eine nützliche Eigenschaft von Äquivalenzen

- **Satz** (Ersetzbarkeitssatz). Ersetzt man in einem v.g.B.A. b einen Teilausdruck durch einen äquivalenten anderen v.g.B.A., so ist der resultierende v.g.B.A. c zu b äquivalent.
- Beispiel: Betrachten wir $\Big(X_1 \wedge \big((\sim X_3) \vee X_4\big)\Big)$.
  Dieser Ausdruck enthält den Teilausdruck $\big((\sim X_3) \vee X_4\big)$.
  Ein zu diesem Teilausdruck äquivalenter Ausdruck ist $\big(X_4 \vee (\sim X_3)\big)$.
  Ersetzt man den Teilausdruck durch den äquivalenten Ausdruck, erhält man ein zu dem Gesamtausdruck äquivalentes Ergebnis:

$$\Big(X_1 \wedge \big((\sim X_3) \vee X_4\big)\Big) \equiv \Big(X_1 \wedge \big(X_4 \vee (\sim X_3)\big)\Big)$$

Beweis des Ersetzbarkeitssatzes

- Beweis durch strukturelle Induktion über den Aufbau von b_1
- Induktionsanfang
  Sei $b_1 \in \{\mathbf{0}, \mathbf{1}\} \cup V$. Somit ist der einzige möglichen Teilausdruck b_2, den b_1 enthält, b_1 selbst, d.h. $b_2 = b_1$. Dann ist aber auch $c_2 = c_1$, und wegen $b_2 \equiv c_2$ folgt $b_1 \equiv c_1$. ■
- Induktionsschritt
  Ist $b_1 = b_2$, folgt die Behauptung mit dem selben Argument wie oben. Sei also im Folgenden $b_1 \neq b_2$.

Beweis des Ersetzbarkeitssatzes (2)

1. Fall. Sei $b = (\sim b')$. Dann ist $c = (\sim c')$, wobei c' durch Ersetzung eines Teilausdrucks in b' durch einen äquivalenten v.g.B.A. entstanden ist. Nach Induktionsvoraussetzung sind b' und c' äquivalent, sodass dann für alle Belegungen β gilt:

$$\phi_\beta(b) = \phi_\beta\big((\sim b')\big) = \sim\Big(\phi_\beta(b')\Big) = \sim\Big(\phi_\beta(c')\Big) = \phi_\beta\big((\sim c')\big) = \phi_\beta(c)$$

Also sind b und c äquivalent. ■

Beweis des Ersetzbarkeitssatzes (3)

2. Fall. Sei $b = (b_1 \wedge b_2)$. Dann ist $c = (c_1 \wedge b_2)$ oder $c = (b_1 \wedge c_2)$, wobei c_1 und c_2 durch Ersetzung eines Teilausdrucks in b_1 bzw. b_2 durch einen jeweils äquivalenten v.g.B.A. hervorgegangen ist. Für ersteren Fall gilt dann nach Induktionsannahme für alle Belegungen β:

$$\phi_\beta(b) = \phi_\beta\big((b_1 \wedge b_2)\big) = \wedge\Big(\phi_\beta(b_1), \phi_\beta(b_2)\Big) = \wedge\Big(\phi_\beta(c_1), \phi_\beta(b_2)\Big)$$
$$= \phi_\beta\big((c_1 \wedge b_2)\big) = \phi_\beta(c)$$

Den Fall für c_2 zeigt man analog. ■

3. Fall. Sei $b_1 = (b_1' \vee b_1'')$. Die Beweisführung erfolgt analog zu Fall 2. ■

Beweis des Ersetzbarkeitssatzes (4)

4. Fall: $b = f(b_1, \ldots, b_k)$. Dann hat c die Form $f(b_1, \ldots, c_i, \ldots, b_k)$, wobei c_i aus b_i durch Ersetzung eines Teilausdruck durch einen äquivalenten v.g.B.A. entstanden ist. Nach Induktionsannahme gilt dann für alle Belegungen β:

$$\begin{aligned}
\phi_\beta(b) &= \phi_\beta\big(f(b_1, \ldots, b_i, \ldots, b_k)\big) \\
&= f\left(\phi_\beta(b_1), \ldots, \phi_\beta(b_i), \ldots, \phi_\beta(b_k)\right) \\
&= f\left(\phi_\beta(b_1), \ldots, \phi_\beta(c_i), \ldots, \phi_\beta(b_k)\right) \\
&= \phi_\beta\big(f(b_1, \ldots, c_i, \ldots, b_k)\big) \\
&= \phi_\beta(c)
\end{aligned}$$

qed

Boole’sche Ausdrücke ohne Funktionssymbole

- Wir wollen nun zeigen, dass man bei v.g. Boole’schen Ausdrücken auch gut auf Funktionssymbole der Form $f(b_1, \ldots, b_k)$ verzichten kann.
- Dazu führen wir zunächst zwei Hilfsdefinitionen ein.
- **Definition.** Sei b ein v.g.B.A. Dann bezeichnet

$$b^{\epsilon} = \begin{cases} (\sim b), & \text{falls } \epsilon = 0 \\ b, & \text{falls } \epsilon = 1 \end{cases}$$

- Beispiel:
  - $\big(X_{42} \vee f(\mathbf{0})\big)^0 = \big(\sim\big(X_{42} \vee f(\mathbf{0})\big)\big)$
  - $(1 \wedge X_2)^1 = (1 \wedge X_2)$

Träger einer Schaltfunktion

- **Definition.** Sei f eine k-stellige Schaltfunktion. Die Menge

$$T(f) = \{a \in \{0,1\}^k \mid f(a) = 1\}$$

heißt der *Träger* von f.

- Beispiel: Sei f die durch die nebenstehende Tabelle definierte dreistellige Schaltfunktion.
  Der Träger von f ist dann
  $T(f) = \{(0,1,0), (1,0,1), (1,1,0)\}$.

x_1	x_2	x_3	$f(x_1, x_2, x_3)$
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	0
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	0

Boole’sche Ausdrücke ohne Funktionssymbole (2)

- **Lemma.** Zu jedem Boole’schen Ausdruck b gibt es einen zu b äquivalenten B.A. c, in dem keine Funktionssymbole vorkommen.

  Beweis per struktureller Induktion über b.

  Induktionsanfang.
  Für $b \in \{\mathbf{0}, \mathbf{1}\} \cup V$ gilt die Behauptung offensichtlich, indem man $c = b$ setzt.

  Induktionsschritt.
  Hat b die Form $(\sim b')$ oder $(b_1 \wedge b_2)$ oder $(b_1 \vee b_2)$ folgt die Behauptung direkt aus der Induktionsannahme für b' bzw. b_1 und b_2.

Boole’sche Ausdrücke ohne Funktionssymbole (3)

Es bleibt der Fall $b = f(b_1, \dots, b_k)$. Sei dazu β eine beliebige Belegung. Dann folgt:

$\phi_\beta\big(f(b_1, \dots, b_k)\big) = 1$

$\Leftrightarrow f\left(\phi_\beta(b_1), \dots, \phi_\beta(b_k)\right) = 1$

$\Leftrightarrow \left(\phi_\beta(b_1), \dots, \phi_\beta(b_k)\right) \in T(f)$

$\Leftrightarrow \phi_\beta\ (b_i) = a_i$ für alle $1 \leq i \leq k$ für ein $a \in T(f)$

$\Leftrightarrow \phi_\beta\big(b_i^{a_i}\big) = 1$ für alle $1 \leq i \leq k$ für ein $a \in T(f)$

$\Leftrightarrow \phi_\beta\left(\bigwedge_{i=1}^{k} b_i^{a_i}\right) = 1$ für ein $a \in T(f)$

$\Leftrightarrow \phi_\beta\left(\bigvee_{a \in T(f)}\left(\bigwedge_{i=1}^{k} b_i^{a_i}\right)\right) = 1$

Nach Induktionsannahme gibt es für jeden Ausdruck b_i einen äquivalenten v.g.B.A. c_i ohne Funktionssymbol.

Somit ist $\bigvee_{a \in T(f)}\big(\bigwedge_{i=1}^{k} c_i^{a_i}\big)$ nach dem Ersetzbarkeitssatz zu b äquivalent und enthält auch keine Funktionssymbole.

qed.

Reflektionen über das Erreichte

1. Funktionsterme in v.g.B.A. sind also nur „syntaktischer Zucker“, die die Ausdrucksmächtigkeit klassischer logischer Formeln nicht erweitern. (Sie können aber dennoch praktisch sein.)
2. Wegen der direkten Verbindung zwischen syntaktischen Funktionssymbolen f in v.g.B.A. und Schaltfunktionen f mittels der Wertefunktion ϕ_β ist mit Blick auf den letzten Fall des Beweises nun legitim zu sagen, dass jede Schaltfunktion durch einen v.g.B.A. ohne Funktionssymbole „berechnet“ werden kann.
3. Die syntaktische Form des letzten Ausdrucks in obigem Beweis ist ansprechend und motiviert die nun folgende Definition.

Normalformen

- **Definition.** Ein Boole'scher Ausdruck b ist in *konjunktiver Normalform* (KNF), falls er eine Konjunktion von Disjunktionen von Literalen, **0** oder **1** ist:

$$b = \bigwedge_{i=1}^{k} \left(\bigvee_{j=1}^{l_i} L_{i,j} \right)$$

  Ein Boole'scher Ausdruck b ist in *disjunktiver Normalform* (DNF), falls er eine Disjunktion von Konjunktionen von Literalen, **0** oder **1** ist:

$$b = \bigvee_{i=1}^{k} \left(\bigwedge_{j=1}^{l_i} L_{i,j} \right)$$

  Es sind also alle $L_{i,j} \in \{X_1, \dots, X_n\} \cup \{\overline{X_1}, \dots, \overline{X_n}\} \cup \{\mathbf{0}, \mathbf{1}\}$.

Warum Normalformen

- Normalformen dienen dazu, die syntaktische Vielfalt von v.g.B.A. auf eine „standardisierte“ Form einzuschränken.
- Dies hat z.B. den Vorteil, dann bei Äquivalenzbeweisen auf viele Fallunterscheidungen verzichten zu können.
- Auch macht es eine Normalform etwas einfacher, über bestimmte Eigenschaften von v.g.B.A. nachzudenken, da die Normalformen strukturell sehr „handlich“ sind.
- Damit dies geht, muss aber zunächst gewährleistet sein, dass es für jeden beliebigen v.g.B.A. einen äquivalenten v.g.B.A. in Normalform gibt.

Äquivalenz von Normalformen

- **Satz.** Zu jedem Boole'schen Ausdruck b_1 gibt es einen Boole'schen Ausdruck b_2, der in KNF ist, und einen Boole'schen Ausdruck b_3, der in DNF ist, sodass gilt: $b_1 \equiv b_2 \equiv b_3$.
- Beweis per struktureller Induktion über b_1.

  Induktionsanfang.

  Ist $b_1 = 0$ oder $b_1 = 1$ oder $b_1 = X_i \in V$, dann ist b_1 selbst bereits in KNF und in DNF. ■

Äquivalenz von Normalformen (2)

Induktionsschritt

Fall 1: b_1 hat die Form $(\sim b_1')$ für einen Boole'schen Ausdruck b_1'. Nach Induktionsannahme gibt es dann einen B.A. b_2' in DNF mit $b_1' \equiv b_2'$. Dann gilt:

$$(\sim b_1') \equiv (\sim b_2') = \left(\sim \bigvee_{i=1}^{k} \left(\bigwedge_{j=1}^{l_i} L_{i,j} \right) \right) \equiv \bigwedge_{i=1}^{k} \left(\sim \bigwedge_{j=1}^{l_i} L_{i,j} \right) \equiv \bigwedge_{i=1}^{k} \left(\bigvee_{j=1}^{l_i} (\sim L_{i,j}) \right)$$

Da $(\sim 0) \equiv 1$, $(\sim 1) \equiv 0$ und $\left(\sim\left(\sim L_{i,j}\right)\right) \equiv L_{i,j}$ gibt es also einen zu $(\sim b_1')$ äquivalenten Ausdruck in KNF. Die Behauptung für b_3' zeigt man analog, indem man alle ∨ durch ∧ ersetzt und umgekehrt. ■

Äquivalenz von Normalformen (3)

Fall 2: b_1 hat die Form $(b_1' \vee b_1'')$. Nach Induktionsannahme gibt es zu b_1' und b_1'' äquivalente Boole'sche Ausdrücke b_3' bzw. b_3'' in DNF. Dann ist $b_3 = b_3' \vee b_3''$ zu b_1 äquivalent und in DNF.
Weiterhin gibt es nach Induktionsannahme b_1' und b_1'' äquivalente Boole'sche Ausdrücke b_2' bzw. b_2'' in KNF, d.h. wir können schreiben: $b_2' = \bigwedge_{i=1}^{k} b_{2,i}'$ und $b_2'' = \bigwedge_{j=1}^{l} b_{2,j}''$ wobei alle $b_{2,i}'$ und $b_{2,j}''$ Disjunktionen sind. Es folgt:

$$(b_1' \vee b_1'') \equiv \left(\left(\bigwedge_{i=1}^{k} b_{2,i}'\right) \vee \left(\bigwedge_{j=1}^{l} b_{2,j}''\right)\right) \overset{(1)}{\equiv} \left(\bigwedge_{i=1}^{k}\left(\bigwedge_{j=1}^{l}\left(b_{2,i}' \vee b_{2,j}''\right)\right)\right) \overset{(2)}{\equiv} \bigwedge_{i=1}^{k\cdot l} b_{1,i}$$

wobei wir bei (1) das Distributivgesetz und bei (2) das Assoziativgesetz angewendet haben und $b_{1,i}$ eine Disjunktion ist. Somit ist der letzte Ausdruck in KNF. ■

Äquivalenz von Normalformen (4)

Fall 3. b_1 hat die Form $(b_1' \wedge b_1'')$. Diesen Fall zeigt man nach demselben Muster wie Fall 2.

Fall 4. b_1 hat die Form $f\left(b_{1,1}, \ldots, b_{1,k}\right)$. Dann gibt es wie zuvor bewiesen einen zu b_1 äquivalenten v.g.B.A., in dem kein Funktionssymbol vorkommt. Dieser ist dann aber bereits durch die Fälle 1-3 abgedeckt. *qed.*

Rückbesinnung

- Wir haben nun ein beachtliches Repertoire an Lemmas und Sätzen sowie syntaktischen Varianten und Äquivalenzen von v.g.B.A. aufgebaut.
- Doch was hilft uns das alles für die Betrachtung logischer Aussagen?
- Viel, wie wir nun sehen werden, wenn wir uns einer zentralen Frage der Aussagenlogik widmen: ergibt eine gegebene Menge logischer Aussagen zusammengenommen einen Sinn, oder enthalten sie einen inhärenten Widerspruch?
- „Sinn ergeben“ ist eine vage Formulierung – was wir (formal gesprochen) meinen, ist: ist die Konjunktion gegebener Aussagen *erfüllbar* oder nicht?

Kalküle

- Ein *Kalkül* ist ein System einfach anzuwendender Regeln, das syntaktische Umformungen ermöglicht.
- Beispielsweise könnte man Äquivalenzen als Umformungsregeln auffassen
  - Wenn es uns etwa leichter fiele, die Nichterfüllbarkeit einer Konjunktion nachzuweisen als die Erfüllbarkeit einer Disjunktion, könnten wir z.B. den gegeben Ausdruck $(X_1 \vee X_2)$ dank De Morgan umformen zu $\sim(\sim X_1 \wedge \sim X_2)$
- Da es sich bei einem Kalkül um rein syntaktische Umformungen handelt, eignet er sich für die Implementierung mit einem Rechner.

Resolutionskalkül

- Wir führen nun einen Kalkül ein, mit dem die *Unerfüllbarkeit* logischer Aussagen automatisch geprüft werden kann.
- Das ist durchaus erstaunlich, da wir ja von rein syntaktischen Umformungen sprechen!
- Ebenso erstaunlich ist, dass wir dafür nur eine einzige Umformungsregel benötigen.
- Weniger erstaunlich ist, dass wir voraussetzen, dass die Aussagen als eine Boole’sche Aussage in KNF vorliegen.

Resolutionskalkül (2)

- Sei b eine Boole'sche Aussage in KNF, d.h. b hat die Form

$$b = \left(L_{1,1} \vee \cdots \vee L_{1,l_1}\right) \wedge \cdots \wedge \left(L_{k,1} \vee \cdots \vee L_{k,l_k}\right)$$

  wobei für alle $L_{i,j}$ gilt: $L_{i,j} \in \{X_1, \ldots, X_n\} \cup \{\overline{X_1}, \ldots, \overline{X_n}\} \cup \{\mathbf{0}, \mathbf{1}\}$.

- Für die Resolution ist es vorteilhaft, B.A., die in KNF vorliegen, als Mengen darzustellen:

$$\left\{\left\{L_{1,1}, \ldots, L_{1,l_1}\right\}, \ldots, \left\{L_{k,1}, \ldots, L_{k,l_k}\right\}\right\}$$

  Die Elemente dieser Menge sind also selbst wieder Mengen, die wir *Klauseln* nennen.

Drei Beobachtungen bgzl. $\mathbf{0}$ und $\mathbf{1}$ in Bezug auf Resolution

1. Enthält eine Klauselmenge F die Klausel $\{\mathbf{0}\}$, dann ist F unerfüllbar.
   ⇒ Klauselmengen, die $\{\mathbf{0}\}$ enthalten, sind trivial auf Unerfüllbarkeit zu testen.
2. Für jede Klausel K gilt: $K \equiv K \cup \{\mathbf{0}\}$
   ⇒ Aus diesem Grund können wir, wenn wir auf Unerfüllbarkeit testen, bei Klauseln, die $\mathbf{0}$ enthalten, diese getrost aus der Klausel streichen.
3. Jede Klausel K mit $\mathbf{1} \in K$ ist erfüllbar.
   Daher gilt für jede Formelmenge F: $F \equiv F \cup K$
   ⇒ Aus diesem Grund können wir, wenn wir auf Unerfüllbarkeit testen, Klauseln, die $\mathbf{1}$ enthalten, aus der Klauselmenge streichen.

Im Folgenden gehen wir daher davon aus, dass alle betrachteten Klauselmengen $\mathbf{0}$ oder $\mathbf{1}$ nicht enthalten.

Resolvent

- **Definition.** Seien K_1, K_2 und R Klauseln. Gilt $L \in K_1$ und $\tilde{L} \in K_2$ und hat R die Form

$$R = (K_1 \backslash \{L\}) \cup \left(K_2 \backslash \{\tilde{L}\}\right)$$

dann heißt R *Resolvent* von K_1 und K_2, wobei L ein Literal ist, und $\tilde{L}$ wie folgt definiert ist:

$$\tilde{L} = \begin{cases} \overline{X}_i, & \text{falls } L = X_i \\ X_i, & \text{falls } L = \overline{X}_i \end{cases}$$

Beispiel (1)

- Seien $K_1 = \{X_3, \overline{X_4}, X_1\}$ und $K_2 = \{X_4, \overline{X_1}\}$.
- Dann gilt für $L = \overline{X_4}$:
  1. $\tilde{L} = X_4$
  2. $L \in K_1$ und $\tilde{L} \in K_2$
- Somit ist $R = (K_1 \backslash \{L\}) \cup \left(K_2 \backslash \{\tilde{L}\}\right)$
  $= (\{X_3, \overline{X_4}, X_1\} \backslash \{\overline{X_4}\}) \cup (\{X_4, \overline{X_1}\} \backslash \{X_4\})$
  $= \{X_3, X_1\} \cup \{\overline{X_1}\}$
  $= \{X_3, X_1, \overline{X_1}\}$

  ein Resolvent von K_1 und K_2.

Quiz

- Seien $K_1 = \{X_5\}$ und $K_2 = \{\overline{X_5}\}$.
  Gibt es für K_1 und K_2 ein Resolvent?
- Ja.
- Für $L = X_5$ ist $\tilde{L} = \overline{X_5}$ und $R = (\{X_5\} \backslash \{X_5\}) \cup (\{\overline{X_5}\} \backslash \{\overline{X_5}\}) = \{\}$

Leere Menge im Resolutionskalkül

- Im Resolutionskalkül verwendet man als Symbol für die leere Menge nicht {}, sondern □.
- Sie bezeichnet eine unerfüllbare Formel.
- Eine Klauselmenge, die □ enthält, ist somit per Definition unerfüllbar.

Resolutionslemma

- **Lemma (Resolutionslemma).** Sei F eine Klauselmenge, die einen v.g.B.A. darstellt. Ferner sei R ein Resolvent zweier Klauseln K_1 und K_2 in F. Dann sind F und $F \cup \{R\}$ äquivalent.
- Beweis. Sei β eine Belegung.
  Falls $\beta \vDash F \cup R$, dann muss natürlich erst recht $\beta \vDash F$ gelten.

  Nehmen wir also umgekehrt an, dass gilt: $\beta \vDash F$. Dann gilt auch $\beta \vDash K$ für alle Klauseln $K \in F$. R habe die Form $R = (K_1 \backslash \{L\}) \cup \left(K_2 \backslash \{\tilde{L}\}\right)$ für ein $L \in K_1$ und $\tilde{L} \in K_2$.

  1. Fall: $\beta \vDash L$
  Dann folgt $\beta \nvDash \tilde{L}$, und wegen $\beta \vDash K_2$ gilt dann $\beta \vDash \left(K_2 \backslash \{\tilde{L}\}\right)$ und somit $\beta \vDash R$.

  2. Fall: $\beta \nvDash L$
  Dann folgt wegen $\beta \vDash K_1$, dass gilt: $\beta \vDash (K_1 \backslash \{L\})$, und damit gilt auch $\beta \vDash R$.

  qed.

Prinzip des Resolutionskalküls

- Die Erkenntnis des vorherigen Satzes zusammen mit der Tatsache, dass Klauselmengen, die □ enthalten, unerfüllbar sind, sind freudige Nachrichten!
- Nun können wir nämlich einfach folgendermaßen vorgehen:
- Gegeben eine Klauselmenge, erzeuge sukzessive und wiederholt alle möglichen Resolventen, bis □ darunter ist.
- Dann weiß man, dass die Ausgangsmenge an Klauseln unerfüllbar war.

Resolutionsdefinition

- **Definition.** Sei F eine Klauselmenge. Dann ist $Res(F)$ definiert als

$$Res(F) = F \cup \{R | R \text{ ist Resolvent zweier Klauseln in } F\}$$

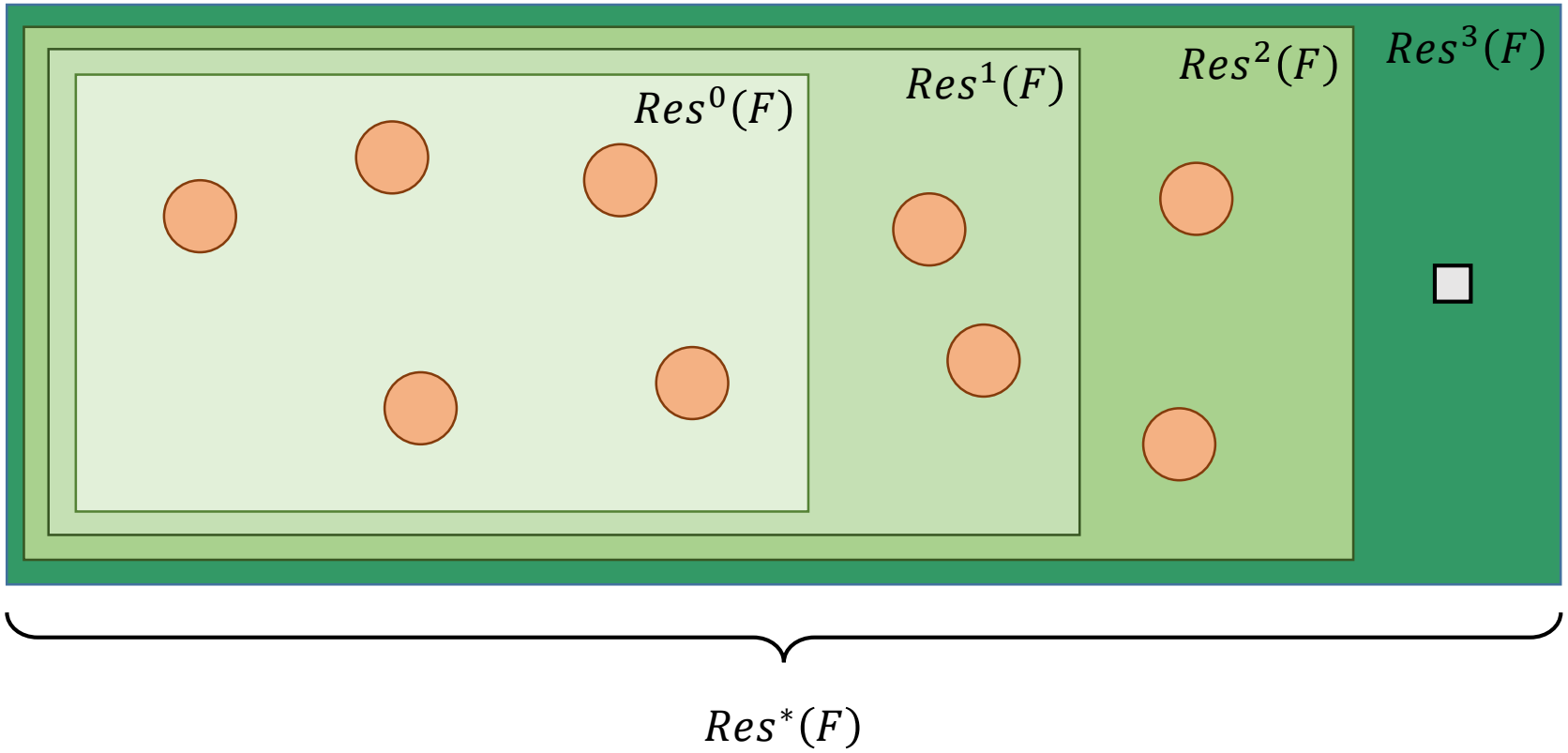
Weiterhin setzen wir:

$$Res^0(F) = F$$
$$Res^{n+1}(F) = Res\left(Res^n(F)\right) \text{ für } n \geq 0$$

und schließlich:

$$Res^*(F) = \bigcup_{n \geq 0} Res^n(F)$$

Veranschaulichung der Idee



Resolutionssatz der Aussagenlogik

- **Satz (Resolutionssatz der Aussagenlogik).** Eine endliche Klauselmenge F ist unerfüllbar genau dann, wenn $\square \in Res^*(F)$.

  Beweis
  1. "$\Leftarrow$" (Korrektheit)
  Sei $\square \in Res^*(F)$. Dann muss es ein $n > 0$ geben, sodass auch $\square \in Res^n(F)$. Dazu müssen aber zwei Klauseln $K_1, K_2 \in Res^{n-1}(F)$ existieren, welche die Form $K_1 = \{L\}$ und $K_2 = \{\tilde{L}\}$ haben. Somit ist $Res^{n-1}(F)$ unerfüllbar. Da nach dem Resolutionslemma gilt:

$$F \equiv Res^1(F) \equiv \cdots \equiv Res^{n-1}(F)$$

  ist dann aber auch F unerfüllbar. ■

Beweis (2)

- 2. “⇒” (Vollständigkeit)
  Sei F eine unerfüllbare Klauselmenge. Wir zeigen, dass dann □ ∈ $Res^*(F)$ gilt, durch vollständige Induktion über die Anzahl m der in F enthaltenen Variablen.
  Induktionsanfang. Sei $m = 1$.
  Um unerfüllbar zu sein, muss F zwei Klauseln der Form $\{X_i\}$ bzw. $\{\overline{X_i}\}$ enthalten. Dann ist aber bereits □ ∈ $Res^1(F)$ ■

Beweis (3)

Induktionsschritt $n \to n+1$

Sei F eine Klauselmenge, in der $n+1$ Variablen vorkommen, die wir der Einfachheit halber mit $X_1, \ldots, X_{n+1}$ bezeichnen. Wir konstruieren daraus zunächst zwei Klauselmengen F_0 und F_1 mit je n Variablen:

F_0 enthalte alle Klauseln von F außer denen, in denen $\overline{X_{n+1}}$ vorkommt. Zusätzlich wird in F_0 noch in jeder Klausel aus F, in der X_{n+1} vorkommt, dieses Literal aus der Klausel gestrichen.

F_1 enthalte alle Klauseln von F außer denen, in denen X_{n+1} vorkommt. Zusätzlich wird in F_1 noch in jeder Klausel aus F, in der $\overline{X_{n+1}}$ vorkommt, dieses Literal aus der Klausel gestrichen.

Um auf F_0 und F_1 die Induktionsannahme anwenden zu können, müssen wir aber außerdem noch nachweisen, dass beide unerfüllbar sind.

Beweis (4)

Angenommen, es gäbe eine Belegung β, die F_0 erfüllt. Dann wäre aber die folgendermaßen definierte Belegung β' ein Modell für F, was im Widerspruch zu der Annahme steht, dass F unerfüllbar ist.

$$\beta'(X) = \begin{cases} 0, & falls\ X = X_{n+1} \\ \beta(X), & sonst \end{cases}$$

Analog argumentiert man, dass F_1 unerfüllbar sein muss.

Somit lässt sich also die Induktionsannahme auf F_0 und F_1 anwenden und es gilt: $\square \in Res^*(F_0)$ und $\square \in Res^*(F_1)$.

Beweis (5)

Per Konstruktion gilt für jedes $K \in F_0$: $K \in F$ oder $(K \cup \{X_{n+1}\}) \in F$. Man sieht schnell ein, dass dann auch für jedes $K \in Res^i(F_0)$, $i \in \mathbb{N}$, gilt: $K \in Res^i(F)$ oder $(K \cup \{X_{n+1}\}) \in Res^i(F)$.

Sei t die kleinste Zahl für die gilt: $\square \in Res^t(F_0)$. Dann muss also gelten: $\square \in Res^t(F)$ oder $\{X_{n+1}\} \in Res^t(F)$. Somit ist dann auch $\square \in Res^*(F)$ oder $\{X_{n+1}\} \in Res^*(F)$.

Analog zeigt man mittels F_1, dass auch $\square \in Res^*(F)$ oder $\{\overline{X_{n+1}}\} \in Res^*(F)$ gelten muss.

Gilt Ersteres, also $\square \in Res^*(F)$, ist man fertig. Andernfalls folgt jedoch dasselbe aus $\{X_{n+1}\} \in Res^*(F)$ und $\{\overline{X_{n+1}}\} \in Res^*(F)$ nach einem weiteren Resolutionsschritt.

qed.

Quiz

- Gilt für jede beliebige Klauselmenge F, dass sich nach Anwendung des Resolutionsverfahrens die leere Klausel $\square \in Res^*(F)$ befindet?
  - Nein – das gilt nur für *unerfüllbare* Klauselmengen F.
- Wie läuft das Resolutionsverfahren ab, wenn es auf eine erfüllbare Klauselmenge F angewendet wird?
  - Für irgendein i gilt: $Res^i(F) = Res^{i+1}(F) = \cdots = Res^*(F)$

Minimierung Boole’scher Ausdrücke

- Das obige Resolutionsverfahren testet die Unerfüllbarkeit eines Boole‘schen Ausdrucks.
  - B.A. in Konjunktiver Normalform: $\bigwedge \bigvee L_{i,j}$
  - Repräsentation: Menge von sog. Klauseln ($\rightarrow \bigvee L_{i,j}$)
- Das folgende Verfahren hat einen anderen Zweck: das Minimieren eines Boole’schen Ausdrucks, d.h. Bestimmen eines äquivalenten Ausdrucks mit einer minimalen Anzahl von Literalen.
  - B.A. in Disjunktiver Normalform: $\bigvee \bigwedge L_{i,j}$
  - Repräsentation: Menge von sog. Mintermen ($\rightarrow \bigwedge L_{i,j}$)

Das Verfahren von Quine-McCluskey

- Zwei Schritte:
  1. Ein tabellarisches Resolutionsverfahren
     - Ergebnis: Kandidaten zur Bildung eines minimalen B.A.
  2. Auswahl einer geeigneten Teilmenge der Kandidaten
     - Ergebnis: ein zum Originalausdruck äquivalenter, aber minimaler B.A.
- Die in Schritt eins gewonnenen Kandidaten heißen *Primimplikanten*.

Ein Beispiel zum Einstieg

- DNF:

$$\overline{X_1 X_2} X_3 \overline{X_4} \vee \overline{X_1 X_2} X_3 X_4 \vee \overline{X_1} X_2 X_3 \overline{X_4} \vee X_1 \overline{X_2} X_3 \overline{X_4} \vee X_1 \overline{X_2} X_3 X_4 \vee X_1 X_2 X_3 X_4$$

X_1	X_2	X_3	X_4	$f(X_1, X_2, X_3, X_4)$
0	0	0	0	0
0	0	0	1	0
0	0	1	0	1
0	0	1	1	1
0	1	0	0	0
0	1	0	1	0
0	1	1	0	1
0	1	1	1	0
1	0	0	0	0
1	0	0	1	0
1	0	1	0	1
1	0	1	1	1
1	1	0	0	0
1	1	0	1	0
1	1	1	0	0
1	1	1	1	1

Ein Beispiel zum Einstieg (2)

Minterme des Originalausdrucks

	X_1	X_2	X_3	X_4	
m_1	0	0	1	0	✓
m_2	0	0	1	1	✓
m_3	0	1	1	0	✓
m_4	1	0	1	0	✓
m_5	1	0	1	1	✓
m_6	1	1	1	1	✓

⋮

Abgeleitete Minterme aus Zeilen der Tabelle, die sich an genau einer Stelle unterscheiden

	X_1	X_2	X_3	X_4	
$m_1 + m_2$	0	0	1	-	✓
$m_1 + m_3$	0	-	1	0	p_1
$m_1 + m_4$	-	0	1	0	✓
$m_2 + m_5$	-	0	1	1	✓
$m_4 + m_5$	1	0	1	-	✓
$m_5 + m_6$	1	-	1	1	p_2
$m_1 + m_2 + m_4 + m_5$	-	0	1	-	p_3

Ein Beispiel zum Einstieg (3)

- $m_1 = \overline{X_1 X_2} X_3 \overline{X_4}$
- $m_2 = \overline{X_1 X_2} X_3 X_4$
- $m_3 = \overline{X_1} X_2 X_3 \overline{X_4}$
- $m_4 = X_1 \overline{X_2} X_3 \overline{X_4}$
- $m_5 = X_1 \overline{X_2} X_3 X_4$
- $m_6 = X_1 X_2 X_3 X_4$

- $p_1 = \overline{X_1} X_3 \overline{X_4}$
- $p_2 = \overline{X_1} X_3 \overline{X_4}$
- $p_2 = \overline{X_2} X_3$

$$f(X_1, X_2, X_3, X_4) = \overline{X_1} X_3 \overline{X_4} \vee \overline{X_1} X_3 \overline{X_4} \vee \overline{X_2} X_3$$

	m_1	m_2	m_3	m_4	m_5	m_6
p_1	•		•			
p_2					•	•
p_3	•	•		•	•	

Quiz

- Was, wenn die Tabelle im letzten Schritt z.B. so aussähe?

	m_1	m_2	m_3	m_4	m_5	m_6
p_1		•				•
p_2	•			•		
p_3	•		•	•		•
p_4				•	•	
p_5			•			•

- Dann wird nur ein Teil der Primimplikanten benötigt:
  - p_1 und p_4 werden benötigt, da nur sie m_2 bzw. m_5 abdecken
  - Anschließend braucht man aber nur noch p_3
  - p_2 und p_5 sind redundant

Das Verfahren von Quine-McCluskey

- 1. Teil (Bestimmung der Primimplikanten)
  - Bei n vorkommende Variablen wird eine Tabelle mit $n + 2$ Spalten erstellt.
  - Die Minterme der gegebenen DNF werden in die ersten Zeilen der Tabelle übertragen; diese erhalten die Namen $m_1, \ldots, m_k$ in der ersten Spalte.
  - Es werden alle Paare von Zeilen gesucht, deren Minterme sich an genau einer Stelle unterscheiden. Für diese wird wie folgt eine neue Zeile eingetragen:
    - Der Name der neuen Zeile setzt sich aus den Namen des Paares zusammen
    - Die identischen Minterm-Werte des Paars werden kopiert; an der Stelle, wo sich die beiden unterscheiden, wird ein Strich eingetragen.

    Außerdem wird bei dem Paar jeweils ein Haken in der letzten Spalte gesetzt.
  - Hat eine Zeile keinen passenden Partner, um ein Paar zu bilden, wird bei dieser statt eines Hakens ein neuer Name p_i in die letzte Spalte eingetragen.

Das Verfahren von Quine-McCluskey

- Das Verfahren wird fortgesetzt, bis keine neuen Zeilen mehr gebildet werden können.
- Die Zeilen ohne Haken kennzeichnen die gefundenen Primimplikanten. Zellen mit 0 oder 1 kennzeichnen die enthaltenen Literale des jeweiligen Primpimplikanten.

- 2. Teil (Auswahl der Primimplikanten)
  - Eine neue Tabelle wird erstellt: die Spalten stehen für die ursprünglichen Minterme, die Zeilen für die Primimplikanten
  - Für jeden Primimplikanten werden in dessen Zeile diejenigen Minterme markiert, aus denen sich sein Name zusammensetzt.
  - Dann sucht man nach einer kleinst-möglichen Menge an Primimplikanten, die zusammen genommen für jede Spalte mindestens einen Eintrag liefern.

Zusammenfassung (1)

- Die rekursiv definierte Sprache der vollständig geklammerte Boole'sche Ausdrücke dient uns zur Formulierung komplexer logischer Aussagen aus atomaren Aussagen $X_1, \ldots, X_n$.
- Neben der *Syntax* von Boole'sche Ausdrücken haben wir auch ihre *Semantik* (nämlich die jeweiligen Wahrheitswerte) definiert:
  - zunächst für atomare Aussagen – dazu dienen Belegungen β
  - dann für daraus zusammengesetzte Aussagen – dazu dient die Funktion ϕ_β
- Somit können wir nun über eine *Entsprechung* zweier B.A. in zwei verschiedenen Kontexten reden:
  - syntaktisch, d.h. auf der Ebene von Zeichenketten: „=“ (Gleichheit)
  - semantisch, d.h. bezügich des Wahrheitswertes: „≡“ (Äquivalenz)

Zusammenfassung (2)

- Wir haben eine Vielzahl von Abkürzungen und notationellen Vereinfachungen eingeführt, die uns beim Schreiben Arbeit sparen.
- Wir haben eine Reihe von Äquivalenzen nachgewiesen, die uns erlauben, Boole'sche Ausdrücke umzuformen – ähnlich wie wir das in der Mathematik (Arithmetik) kennen.
  - Assoziativgesetz, Distributivgesetz, Kommutativgesetz, De Morgan-Regeln, …
- Wir haben zwei Normalformen kennengelernt: KNF und DNF.
- Mit der Resolution haben wir ein Verfahren kennengelernt, dass eine Menge von Klauseln automatisch auf Unerfüllbarkeit testet.
- Das Verfahren von Quine-McCluskey findet eine minimale DNF-Repräsentation.