TU WIEN Informatics

06 – Aussagenlogik

Grundzüge digitaler Systeme (192.134)

Vortrag von: Stefan Neumann

Fakultät für Informatik
TU Wien
E-Mail: gds@list.tuwien.ac.at

Aussagenlogik – Übersicht

Klassische Aussagenlogik (Propositionallogik)

- Zwei Wahrheitswerte: wahr/falsch, true/false, verum/falsum, 1/0, ein/aus, ...
- Aussagenvariablen, die wahr oder falsch sein können
- Elementare Operatoren wie „und“, „oder“, „nicht“, ...
- Keine Quantoren

Geht zurück auf die Antike

Grundlegend für

- Philosophie
- Mathematik
- Informatik

Clipart courtesy FCIT

Aristoteles
384–322 v.Chr.

Negation

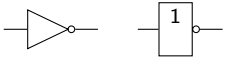
Ich gehe nicht ins Kino.

Ist falsch, wenn ich ins Kino gehe, und wahr andernfalls.

x	not x
0	1
1	0

Andere Bezeichnung: non

Symbole: $\neg x$, $-x$, $\sim x$, x', $!x$, $\overline{x}$, $\mathrm{N}x$, ...

Logikgatter: 

Konjunktion

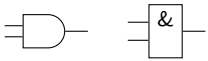
Der Himmel ist blau und die Sonne scheint.

Trifft nur zu, wenn jede der beiden Teilaussagen wahr ist.

x	y	x and y
0	0	0
0	1	0
1	0	0
1	1	1

Andere Bezeichnung: et

Symbole: $x \wedge y$, $x \cdot y$, xy, $x\&y$, $\mathsf{K}xy$, ...

Logikgatter: 

Disjunktion, Alternative

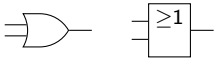
Ich trinke zum Essen Wein oder Bier (oder auch beides).

Nur falsch, wenn ich weder Wein noch Bier trinke.

x	y	x or y
0	0	0
0	1	1
1	0	1
1	1	1

Andere Bezeichnung: vel

Symbole: $x \vee y$, $x + y$, $x \mid y$, Axy

Logikgatter: 

Ausschließende Disjunktion (Antivalenz)

Ich bin entweder gut drauf oder saugrantig,
etwas anderes gibt es bei mir nicht.

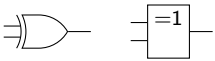
Trifft zu, wenn ich in genau einer der Stimmungslagen bin (die sich ausschließen).

x	y	x xor y
0	0	0
0	1	1
1	0	1
1	1	0

Andere Formulierungen: x oder y

Andere Bezeichnungen: exor, aut

Symbole: $x \not\Leftrightarrow y$, $x \not\equiv y$, $x \oplus y$, $x \nleftrightarrow y$, Jxy, ...

Logikgatter: 

Äquivalenz

Ich springe dann (und nur dann), wenn du es auch tust.

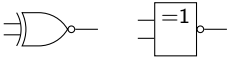
Trifft zu, wenn beide springen oder keiner.

x	y	x iff y
0	0	1
0	1	0
1	0	0
1	1	1

Andere Formulierungen: x genau dann wenn y, x if and only if y, x ist notwendig und hinreichend für y, x ist äquivalent zu y

Andere Bezeichnungen: eq, äq, xnor, nxor, ...

Symbole: $x \Leftrightarrow y$, $x \equiv y$, $x \leftrightarrow y$, Exy, ...

Logikgatter: 

Implikation

Wenn/Falls ich ins Kino gehe, (dann) esse ich dort Popcorn.
Ich gehe nur dann ins Kino, wenn ich dort Popcorn esse.

Falsch, wenn ich im Kino kein Popcorn esse, und wahr, wenn doch.
Keine Festlegung betreffend Popcorn außerhalb des Kinos, daher wahr.

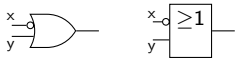
x	y	x implies y
0	0	1
0	1	1
1	0	0
1	1	1

„Verum ex quodlibet“: x implies $1 = 1$
„Ex falso quodlibet“: 0 implies $y = 1$

Andere Formulierungen: aus x folgt y, x impliziert y, x hinreichend für y

Andere Bezeichnung: seq (sequi)

Symbole: $x \Rightarrow y$, $x \supset y$, $x \to y$, Cxy, ...

Logikgatter: 

Implikation (Umkehrung)

Ich esse (dann) Popcorn, wenn/falls ich ins Kino gehe.

x	y	x if y
0	0	1
0	1	0
1	0	1
1	1	1

Andere Formulierungen: x folgt aus y, x wird von y impliziert, x ist notwendig für y

Symbole: $x \Leftarrow y$, $x \subset y$, $x \leftarrow y$, ...

Logikgatter: 

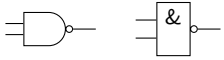
Negierte Konjunktion

x	y	x and y	x nand y
0	0	0	1
0	1	0	1
1	0	0	1
1	1	1	0

Andere Bezeichnungen: Sheffer-Strich, nd (J.Nicod)

Symbole: $x \uparrow y$, $x \mid y$, x/y, Dxy, ...

Logikgatter:



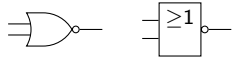
Negierte Disjunktion

x	y	x or y	x nor y
0	0	0	1
0	1	1	0
1	0	1	0
1	1	1	0

Andere Bezeichnungen: Peirce-Pfeil, sh (H.M.Sheffer)

Symbole: $x \downarrow y$, Xxy, ...

Logikgatter:



Wenn Feiertag ist oder der Vortragende krank ist,
findet die Vorlesung nicht statt.

Logische Struktur: „Wenn x oder y, dann nicht z.“

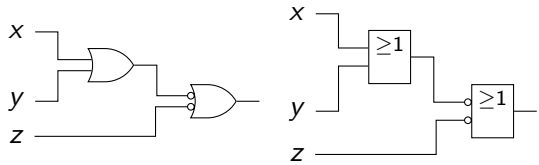
Logische Funktion: (x or y) implies not z implies (or (x, y), not(z))

Logische Formel: $(x \vee y) \Rightarrow \neg z$

... in Prefix-Notation: C A x y N z

... in algebraischer Notation: $(x + y) \rightarrow -z$ $\overline{x}\,\overline{y} + \overline{z}$

Logischer Schaltkreis:



Implikation oder Äquivalenz?

Natürlichsprachliche Implikationen sind oft logische Äquivalenzen.

Wenn du mein Auto wäschst, bekommst du 10 Euro.

Und was, wenn ich es nicht tue?
Nur ein Logiker hält in diesem Fall 10 Euro für möglich.
Alle anderen interpretieren den Satz als:

Du bekommst 10 Euro dann und nur dann, wenn du das Auto wäscht.

Der positive Erfolg bei allen Lehrveranstaltungen und Prüfungen der Studieneingangs- und Orientierungsphase berechtigt zur Absolvierung der weiteren Lehrveranstaltungen und Prüfungen sowie zum Verfassen der im Curriculum vorgesehenen Bachelor- oder Diplomarbeiten.

Universitätsgesetz 2002, Stand Bgbl I Nr. 50/2024, § 66(2)

Logiker: Keine Einschränkung bei nicht bestandener STEOP.
Ministerium: Restliches Studium dann und nur dann, wenn STEOP.

Der Besitz eines Führerscheins berechtigt zum Lenken eines Autos.

Ohne Führerschein keine Berechtigung? (Äquivalenz)
Auch nicht auf Privatgelände? (Doch nur Implikation?)

In formalen Kontexten wird strikt zwischen Implikation und Äquivalenz unterschieden.
„Implikation = halbe Äquivalenz“

Wenn eine Zahl durch 4 teilbar ist, ist sie gerade.

4-Teilbarkeit ist eine hinreichende Bedingung für Geradheit,
aber keine notwendige.

Äquivalenz führt zu einer falschen Aussage:

Eine Zahl ist durch 4 teilbar genau dann, wenn sie gerade ist.

2 ist eine gerade Zahl, aber nicht durch 4 teilbar.

Inklusive oder exklusive Disjunktion?

Natürlichsprachliche Disjunktionen sind meist ausschließend gemeint.

Falls du mich suchst: Ich bin zu Hause oder in der Arbeit.

Physikalisch kann ein Körper nicht an zwei Orten gleichzeitig sein.
Andererseits: Bei Home-Office kann das Büro Teil der Wohnung sein.

„Tee oder Kaffee?“ – „Beides, bitte!“

Eher unüblich, aber der Gast ist König.

Ich besuche dich morgen oder übermorgen.

Ein Besuch morgen *und* übermorgen wäre unerwartet.

Ich fahre entweder Auto oder höre Musik.
(Auf beides gleichzeitig kann ich mich nicht konzentrieren.)

Wirklich ein Beispiel für ausschließende Disjunktion?

Habe ich außerhalb des Autos tatsächlich keine ruhige Minute?

Die exklusive Disjunktion ist hier als Implikation gemeint (und wird auch so verstanden):

Wenn ich Auto fahre, höre ich nicht Musik.

Legt nicht fest, was ich außerhalb des Autos mache.

Rezept für Zweifelsfälle der aussagenlogischen Modellierung

1. Identifiziere die elementaren Aussagen.
2. Analysiere *alle* Wahrheitsbelegungen.
3. Wähle geeignete logische Funktionen
   (unbeirrt von Intuition und natürlicher Sprache).

$z =$ Entweder „ich fahre Auto“ (x) oder „ich höre Musik“ (y).

x	y	z	x or y	x xor y	x implies not y	x nand y
0	0	1	0	0	1	1
0	1	1	1	1	1	1
1	0	1	1	1	1	1
1	1	0	1	0	0	0

x implies not y: Wenn ich Auto fahre, dann höre ich nicht Musik.

x nand y: Es kommt nicht vor, dass ich Auto fahre und Musik höre.

Aussagenlogische Funktionen

false	true
$f_{0,0}$	$f_{0,1}$
0	1
$\bot$	$\top$

	not
x	$f_{1,2}$
0	1
1	0
	$\neg$

		and	nand	or	nor	iff	xor	implies	if		
x	y	$f_{2,1}$	$f_{2,14}$	$f_{2,7}$	$f_{2,8}$	$f_{2,9}$	$f_{2,6}$	$f_{2,13}$	$f_{2,2}$	$f_{2,11}$	$f_{2,4}$
0	0	0	1	0	1	1	0	1	0	1	0
0	1	0	1	1	0	0	1	1	0	0	1
1	0	0	1	1	0	0	1	0	1	1	0
1	1	1	0	1	0	1	0	1	0	1	0
		$\wedge$	$\uparrow$	$\vee$	$\downarrow$	$\Leftrightarrow$	$\not\Leftrightarrow$	$\Rightarrow$	$\not\Rightarrow$	$\Leftarrow$	$\not\Leftarrow$

$f_{n,k}$: systematische Bezeichnung aussagenlogischer Funktionen

- n: Anzahl der Argumente, Stelligkeit
- $k = (f(0,\ldots,0)\, f(0,\ldots,1) \cdots f(1,\ldots,0)\, f(1,\ldots,1))_2$

x	y	x implies y
0	0	1 (msb)
0	1	1
1	0	0
1	1	1 (lsb)

$k = (1101)_2 = 13 \Rightarrow \text{implies} = f_{2,13}$

Aussagenlogische Funktionen

2 0-stellige Funktionen (Konstanten): true, false

4 1-stellige Funktionen: „not“ und drei weniger originelle Funktionen:

- einstellige Konstanten $f_{1,0}$ und $f_{1,3}$ $\quad f_{1,0}(x) = 0,\ f_{1,3}(x) = 1$
- Identitätsfunktion $f_{1,1}$ $\quad f_{1,1}(x) = x$

16 2-stellige Funktionen: and, nand, or, ... von vorhin und 6 weniger originelle Funktionen:

- zweistellige Konstanten $f_{2,0}$ und $f_{2,15}$ $\quad f_{2,0}(x,y) = 0,\ f_{2,15}(x,y) = 1$
- Projektionen $f_{2,3}$ und $f_{2,5}$ $\quad f_{2,3}(x,y) = x,\ f_{2,5}(x,y) = y$
- negierte Projektionen $f_{2,12}$ und $f_{2,10}$ $\quad f_{2,12}(x,y) = \text{not}\, x,\ f_{2,10}(x,y) = \text{not}\, y$

Es gibt 2^{2^n} verschiedene n-stellige logische (Boolesche) Funktionen.

- 2^n verschiedene Argumentkombinationen („Zeilen“)
- 2 Ergebnismöglichkeiten für jede Argumentkombination

Funktionale Vollständigkeit

Eine Menge von Funktionen heißt vollständig (für eine Funktionsmenge),
wenn damit *alle* Funktionen (der Menge) ausgedrückt werden können.

$\{\text{not}, \text{and}, \text{or}\}$ ist funktional vollständig (für die aussagenlogischen Funktionen).

Begründung folgt später.

$\{\text{not}, \text{and}\}$ ist funktional vollständig.

- $\{\text{not}, \text{and}, \text{or}\}$ ist vollständig (siehe oben).
- or kann durch $\{\text{not}, \text{and}\}$ ausgedrückt werden kann:
  $x \text{ or } y = \text{not}(\text{not } x \text{ and not } y)$

x	y	not x	not y	not x and not y	not(not x and not y)
1	1	0	0	0	1
1	0	0	1	0	1
0	1	1	0	0	1
0	0	1	1	1	0

$\{\text{nand}\}$ ist funktional vollständig.

- $\{\text{not}, \text{and}\}$ ist funktional vollständig (siehe oben).
- $\text{not}\, x = x \text{ nand } x$
- $x \text{ and } y = \text{not}(x \text{ nand } y) = (x \text{ nand } y) \text{ nand } (x \text{ nand } y)$

Praktisch relevant: nand ist einfach als Halbleiter-Schaltkreis realisierbar.
Somit ist jede logische Funktion als Schaltkreis realisierbar.

Die Mengen $\{\text{nor}\}$, $\{\text{not}, \text{or}\}$, $\{\text{not}, \text{implies}\}$ und $\{\text{implies}, \text{false}\}$ sind ebenfalls funktional vollständig.

Aussagenlogik – Übersicht

Funktionen und Formeln

Funktionen:
and, nand ... mathematische Funktionen

x and y
$(x \text{ nand } y) \text{ nand } (x \text{ nand } y)$ } ... ununterscheidbar, weil identische Funktion:

x	y	x and y	$(x \text{ nand } y) \text{ nand } (x \text{ nand } y)$
1	1	1	1
1	0	0	0
0	1	0	0
0	0	0	0

Formeln: Für Aussagen über die Form der Ausdrücke benötigen wir eine Formelsprache.

$x \wedge y$
$(x \uparrow y) \uparrow (x \uparrow y)$ } ... unterschiedliche Zeichenketten mit $\begin{matrix}3\\11\end{matrix}$ Symbolen

Syntax und Semantik

Syntax:

- Zeichenfolge (String), mit der etwas notiert wird
- Regeln dafür, welche Zeichenfolgen zulässig sind

Semantik:

- Bedeutung einer Zeichenfolge
- Funktion, die jeder zulässigen Zeichenfolge eine Bedeutung zuordnet

Syntax und Semantik sind grundsätzlich voneinander unabhängig.
Die Bedeutung von Zeichen muss explizit vereinbart werden.

Syntax	Semantik
eins (deutsch) one (englisch) 1 (mathematisch)	das abstrakte Konzept der Zahl „1“
Bug	Schiffsvorderteil (deutsch) Käfer, Programmfehler (englisch)

Syntax und Semantik – Zeichen und Bedeutung

René Magritte, 1929: „Dies ist keine Pfeife“ (sondern das Bild einer solchen)

Induktive Definition unendlicher Mengen

Stufenweise Konstruktion der geraden Zahlen:

- 0 ist eine gerade Zahl: $G_0 = \{0\}$
- Addiert man zu geraden Zahlen 2, erhält man wieder gerade Zahlen:
  $G_1 = G_0 \cup \{n+2 \mid n \in G_0\} = \{0, 2\}$
  $G_2 = G_1 \cup \{n+2 \mid n \in G_1\} = \{0, 2, 4\}$
  $G_{i+1} = G_i \cup \{n+2 \mid n \in G_i\} = \{0, 2, 4, \ldots, 2(i+1)\}$
- Die geraden Zahlen sind alle so konstruierten Zahlen:
  $\mathbb{G} = \lim_{i\to\infty} G_i = \bigcup_{i\geq 0} G_i$

Umständlich, aber konstruktiv: Beginnend mit G_0 lassen sich systematisch alle geraden Zahlen berechnen.

Beobachtung:

- $G_0 \subseteq \mathbb{G}$
- Wenn $n \in \mathbb{G}$, dann auch $n+2 \in \mathbb{G}$.
- $\mathbb{G}$ ist die kleinste Menge mit diesen beiden Eigenschaften.

Induktive Definition der geraden Zahlen

$\mathbb{G}$ ist die kleinste Menge, für die gilt:

- $0 \in \mathbb{G}$
- Wenn $n \in \mathbb{G}$, dann auch $n + 2 \in \mathbb{G}$.

Kompakte Definition, aber nicht konstruktiv:
In den Bedingungen kommt die zu definierende Menge $\mathbb{G}$ selbst vor.

Beide Methoden definieren dieselbe Menge.
(Nicht offensichtlich, Beweis erforderlich!)

Daher: „Use the best of both worlds.“

- Definiere die Menge induktiv.
- Konstruiere benötigte Elemente stufenweise.

Anmerkung: Der Zusatz „ist kleinste Menge“ ist wesentlich.
Die natürlichen Zahlen erfüllen ebenfalls beide Bedingungen, sind aber nicht die kleinste derartige Menge.

Induktive Definition – allgemeine Situation

$\mathcal{U}$ ... Universum, Grundmenge aller möglichen Elemente
$\mathcal{M}_0 \subseteq \mathcal{U}$ ... Menge von Grundelementen
$f_1 : \mathcal{U}^{n_1} \to \mathcal{U}$, $f_2 : \mathcal{U}^{n_2} \to \mathcal{U}$, ... Konstruktionsfunktionen

Stufenweise Konstruktion der Menge $\mathcal{M}$

- $\mathcal{M}_{i+1} = \mathcal{M}_i \cup \{f_1(x_1, \ldots, x_{n_1}) \mid x_1, \ldots, x_{n_1} \in \mathcal{M}_i\}$
  $\cup \{f_2(x_1, \ldots, x_{n_2}) \mid x_1, \ldots, x_{n_2} \in \mathcal{M}_i\}$
  $\cup \cdots$
- $\mathcal{M} = \lim_{i \to \infty} \mathcal{M}_i = \bigcup_{i \geq 0} \mathcal{M}_i$

Induktive Definition der Menge $\mathcal{M}$

$\mathcal{M}$ ist die kleinste Menge, für die gilt:

- $\mathcal{M}_0 \subseteq \mathcal{M}$
- Wenn $x_1, \ldots, x_{n_1} \in \mathcal{M}$, dann $f_1(x_1, \ldots, x_{n_1}) \in \mathcal{M}$.
- Wenn $x_1, \ldots, x_{n_2} \in \mathcal{M}$, dann $f_2(x_1, \ldots, x_{n_2}) \in \mathcal{M}$.
- ...

Aussagenlogik – Syntax

Ausdrücke wie „x and y“ und „$(x \text{ nand } y) \text{ nand } (x \text{ nand } y)$“ repräsentieren dieselbe Funktion.
Um Aussagen über ihre Form treffen zu können, benötigen wir eine Formelsprache.

Aussagenlogische Variablen

$\mathcal{V} = \{A, B, C, \ldots, A_0, A_1, \ldots, x, y, z, x_0, x_1, \ldots\}$

Syntax aussagenlogischer Formeln

Die Menge $\mathcal{A}$ der aussagenlogischen Formeln ist die kleinste Menge, für die gilt:

(a1) $\mathcal{V} \subseteq \mathcal{A}$ — Variablen sind Formeln.

(a2) $\{\top, \bot\} \subseteq \mathcal{A}$ — $\top$ und $\bot$ sind Formeln.

(a3) $\neg F \in \mathcal{A}$, wenn $F \in \mathcal{A}$. — $\neg F$ ist eine Formel, falls F eine ist.

(a4) $(F * G) \in \mathcal{A}$, wenn $F, G \in \mathcal{A}$ und $* \in \{\wedge, \uparrow, \vee, \downarrow, \Leftrightarrow, \not\Leftrightarrow, \Rightarrow, \Leftarrow\}$.
$(F * G)$ ist eine Formel, falls F und G welche sind und $*$ ein binäres Op.symbol ist.

(a1) $\mathcal{V} \subseteq \mathcal{A}$

(a2) $\{\top, \bot\} \subseteq \mathcal{A}$

(a3) $\neg F \in \mathcal{A}$, wenn $F \in \mathcal{A}$.

(a4) $(F * G) \in \mathcal{A}$, wenn $F, G \in \mathcal{A}$ und $* \in \{\wedge, \uparrow, \vee, \downarrow, \Leftrightarrow, \not\Leftrightarrow, \Rightarrow, \Leftarrow\}$.

$((A \uparrow B) \uparrow (A \uparrow B))$ ist eine aussagenlogische Formel, weil:

1. A und B sind Formeln. (a1)
2. $(A \uparrow B)$ ist eine Formel, (a4)
   - da A und B Formeln sind (Punkt 1)
   - und $\uparrow$ ein binäres Operatorsymbol ist.
3. $((A \uparrow B) \uparrow (A \uparrow B))$ ist eine Formel, (a4)
   - da $(A \uparrow B)$ und $(A \uparrow B)$ Formeln sind (Punkt 2)
   - und $\uparrow$ ein binäres Operatorsymbol ist.

(a1) $\mathcal{V} \subseteq \mathcal{A}$

(a2) $\{\top, \bot\} \subseteq \mathcal{A}$

(a3) $\neg F \in \mathcal{A}$, wenn $F \in \mathcal{A}$.

(a4) $(F * G) \in \mathcal{A}$, wenn $F, G \in \mathcal{A}$ und $* \in \{\wedge, \uparrow, \vee, \downarrow, \Leftrightarrow, \not\Leftrightarrow, \Rightarrow, \Leftarrow\}$.

$A \wedge B$ ist keine aussagenlogische Formel.

- $\mathcal{A}$ ist die kleinste Menge mit den Eigenschaften (a1)–(a4), daher kann $\wedge$ nur aufgrund von (a4) in einer Formel vorkommen.
- Dann muss es aber auch ein Klammernpaar geben.
- $A \wedge B$ enthält $\wedge$, aber keine Klammern – Widerspruch.

Formelsyntax: Beispiel einer *induktiven Definition*

$\mathcal{A}$ ist die kleinste Menge, für die gilt:

(a1) $\mathcal{V} \subseteq \mathcal{A}$

(a2) $\{\top, \bot\} \subseteq \mathcal{A}$

(a3) $\neg F \in \mathcal{A}$, wenn $F \in \mathcal{A}$.

(a4) $(F * G) \in \mathcal{A}$, wenn $F, G \in \mathcal{A}$ und $* \in \{\wedge, \uparrow, \vee, \downarrow, \Leftrightarrow, \not\Leftrightarrow, \Rightarrow, \Leftarrow\}$.

$\mathcal{U}$ ... Menge aller Zeichenketten bestehend aus Variablen, Operatorsymbolen und Klammern

$\mathcal{V} \cup \{„\top“, „\bot“\}$ ... Grundelemente

$$\left.\begin{array}{l} f_{\neg}(F) = „\neg“\ F \\ f_{\wedge}(F,G) = „(“\ F\ „\wedge“\ G\ „)“ \\ f_{\uparrow}(F,G) = „(“\ F\ „\uparrow“\ G\ „)“ \\ \quad\vdots \\ f_{\Leftarrow}(F,G) = „(“\ F\ „\Leftarrow“\ G\ „)“ \end{array}\right\} \ldots \text{ Konstruktionsfunktionen}$$

Aussagenlogik – Semantik

$((A \wedge \neg B) \Rightarrow \bot)$ – wahr oder falsch?

Hängt ab

- vom Wert der Variablen A und B und
- von der Bedeutung der Symbole $\wedge$, $\neg$, $\Rightarrow$ und $\bot$.

Interpretationen

$\mathbb{B} = \{0, 1\}$ ... Wahrheitswerte
$I \colon \mathcal{V} \to \mathbb{B}$ ... Wahrheitsbelegung, Interpretation
$\mathcal{I} = \{I \mid I \colon \mathcal{V} \to \mathbb{B}\}$ ... Menge aller Interpretationen

$I(A) = I(C) = 1$ und $I(v) = 0$ sonst
„Die elementaren Aussagen A und C sind wahr, die übrigen sind falsch.“

Semantik aussagenlogischer Formeln

Der Wert einer Formel in einer Interpretation I wird festgelegt durch die Funktion $\mathrm{val}\colon \mathcal{I} \times \mathcal{A} \to \mathbb{B}$:

(v1) $\mathrm{val}_I(A) = I(A)$ für $A \in \mathcal{V}$;

(v2) $\mathrm{val}_I(\top) = 1$ und $\mathrm{val}_I(\bot) = 0$;

(v3) $\mathrm{val}_I(\neg F) = \mathrm{not}\,\mathrm{val}_I(F)$;

(v4) $\mathrm{val}_I(\,(F * G)\,) = \mathrm{val}_I(F) \circledast \mathrm{val}_I(G)$,
wobei $\circledast$ die logische Funktion zum Operator $*$ ist.

(v4) ist eine Abkürzung für:

$$\begin{aligned}
\mathrm{val}_I(\,(F \wedge G)\,) &= \mathrm{val}_I(F)\ \mathrm{and}\ \mathrm{val}_I(G)\\
\mathrm{val}_I(\,(F \vee G)\,) &= \mathrm{val}_I(F)\ \mathrm{or}\ \mathrm{val}_I(G)\\
\mathrm{val}_I(\,(F \Leftrightarrow G)\,) &= \mathrm{val}_I(F)\ \mathrm{iff}\ \mathrm{val}_I(G)\\
\mathrm{val}_I(\,(F \Rightarrow G)\,) &= \mathrm{val}_I(F)\ \mathrm{implies}\ \mathrm{val}_I(G)\\
&\vdots
\end{aligned}$$

Beispiel: Formelauswertung mittels val_I

(v1) $\mathrm{val}_I(A) = I(A)$ für $A \in \mathcal{V}$;

(v2) $\mathrm{val}_I(\top) = 1$ und $\mathrm{val}_I(\bot) = 0$;

(v3) $\mathrm{val}_I(\neg F) = \mathrm{not}\,\mathrm{val}_I(F)$;

(v4) $\mathrm{val}_I(\,(F * G)\,) = \mathrm{val}_I(F) \circledast \mathrm{val}_I(G)$,
wobei $\circledast$ die logische Funktion zum Operator $*$ ist.

Wert von $((A \wedge \neg B) \Rightarrow \bot)$ für $I(A) = 1$ und $I(B) = 0$

$$
\begin{aligned}
\mathrm{val}_I(((A \wedge \neg B) \Rightarrow \bot)) &= \mathrm{val}_I((A \wedge \neg B)) \text{ implies } \mathrm{val}_I(\bot) \\
&= (\mathrm{val}_I(A) \text{ and } \mathrm{val}_I(\neg B)) \text{ implies } 0 \\
&= (1 \text{ and not } \mathrm{val}_I(B)) \text{ implies } 0 \\
&= (1 \text{ and not } 0) \text{ implies } 0 \\
&= (1 \text{ and } 1) \text{ implies } 0 \\
&= 1 \text{ implies } 0 = 0
\end{aligned}
$$

Wahrheitstabelle

- Kompakte Berechnung der Formelwerte für alle Interpretationen
- Unter jedem Operator steht der Wert der entsprechenden Teilformel.
  (Praxis: einfache Spalten können fehlen, „im Kopf“ rechnen)

A	B	$((A \wedge \neg B) \Rightarrow \bot)$
0	0	0 0 1 1 0 1 0
0	1	0 0 0 0 1 1 0
1	0	1 1 1 1 0 0 0
1	1	1 0 0 1 1 1 0

bedeutet:

$I(A) = 0$, $I(B) = 0$: $\mathrm{val}_I(\cdots) = \cdots = 1$

$I(A) = 0$, $I(B) = 1$: $\mathrm{val}_I(\cdots) = \cdots = 1$

$I(A) = 1$, $I(B) = 0$: $\mathrm{val}_I(\cdots) = \cdots = 0$

$I(A) = 1$, $I(B) = 1$: $\mathrm{val}_I(\cdots) = \cdots = 1$

false
0
$\bot$

x	not
0	1
1	0
	$\neg$

x	y	and	implies
0	0	0	1
0	1	0	1
1	0	0	0
1	1	1	1
		$\wedge$	$\Rightarrow$

Klassifikation von Formeln

Eine Formel F heißt

- *gültig*, wenn $\text{val}_I(F) = 1$ für alle $I \in \mathcal{I}$; „Tautologie“
- *erfüllbar*, wenn $\text{val}_I(F) = 1$ für mindestens ein $I \in \mathcal{I}$;
- *widerlegbar*, wenn $\text{val}_I(F) = 0$ für mindestens ein $I \in \mathcal{I}$;
- *unerfüllbar*, wenn $\text{val}_I(F) = 0$ für alle $I \in \mathcal{I}$. „Kontradiktion“

Folgerungen:

- Gültige Formel sind erfüllbar, aber weder widerlegbar noch unerfüllbar.
- Erfüllbare Formel können auch gültig oder widerlegbar sein, aber nie unerfüllbar.
- Widerlegbare Formel können auch erfüllbar oder unerfüllbar sein, aber nie gültig.
- Unerfüllbare Formel sind widerlegbar, aber weder gültig noch erfüllbar.
- F ist gültig/erfüllbar/widerlegbar/unerfüllbar genau dann, wenn $\neg F$ unerfüllbar/widerlegbar/erfüllbar/gültig ist.

$((A \wedge \neg B) \Rightarrow \bot)$ ist erfüllbar und widerlegbar.

A	B	$((A \wedge \neg B) \Rightarrow \bot)$
0	0	0 0 1 0 1 0
0	1	0 0 0 1 1 0
1	0	1 1 1 0 0 0
1	1	1 0 0 1 1 0

Die Formel ist

- erfüllbar (daher nicht unerfüllbar),
- widerlegbar (daher nicht gültig).

$(A \vee \neg A)$ ist gültig und erfüllbar.

A	$(A \vee \neg A)$
0	0 1 1 0
1	1 1 0 1

Die Formel ist

- gültig (daher nicht widerlegbar),
- erfüllbar (daher nicht unerfüllbar).

$(A \wedge \neg A)$ ist unerfüllbar und widerlegbar.

A	$(A \wedge \neg A)$
0	0 0 1 0
1	1 0 0 1

Die Formel ist

- unerfüllbar (daher nicht erfüllbar),
- widerlegbar (daher nicht gültig).

Semantische Äquivalenz 1/2

Semantische Äquivalenz

Zwei Formeln F und G heißen *äquivalent*, geschrieben $F = G$,
wenn $\mathrm{val}_I(F) = \mathrm{val}_I(G)$ für alle Interpretationen I gilt.

$\neg(A \wedge B)$ und $(\neg A \vee \neg B)$ sind äquivalent

A	B	$\neg(A \wedge B) = (\neg A \vee \neg B)$
0	0	1 0 0 0 ✓ 1 0 1 1 0
0	1	1 0 0 1 ✓ 1 0 1 0 1
1	0	1 1 0 0 ✓ 0 1 1 1 0
1	1	0 1 1 1 ✓ 0 1 0 0 1

Semantische Äquivalenz 2/2

Äquivalenz bleibt bei der Ersetzung von Variablen durch Formeln erhalten.

$\neg(\quad A \quad \wedge\ B\ ) = (\neg \quad A \quad \vee \neg\ B\ )$ $\qquad A \mapsto (C \vee D),\ B \mapsto \neg D$

Ersetzen einer Teilformel durch eine äquivalente liefert eine äquiv. Formel.

$(A \Rightarrow \neg(A \wedge B))$ $\qquad \neg(A \wedge B) = (\neg A \vee \neg B)$

Semantische Äquivalenz 2/2

Äquivalenz bleibt bei der Ersetzung von Variablen durch Formeln erhalten.

$\neg((C \vee D) \wedge \neg D) = (\neg(C \vee D) \vee \neg\neg D)$ $\quad A \mapsto (C \vee D),\ B \mapsto \neg D$

Ersetzen einer Teilformel durch eine äquivalente liefert eine äquiv. Formel.

$(A \Rightarrow \neg(A \wedge B)) = (A \Rightarrow (\neg A \vee \neg B))$ $\quad \neg(A \wedge B) = (\neg A \vee \neg B)$

Logische Konsequenz

$F_1, \ldots, F_n \models_I G$: „Aus $\mathrm{val}_I(F_1) = \cdots = \mathrm{val}_I(F_n) = 1$ folgt $\mathrm{val}_I(G) = 1$.“

„Falls in der Wahrheitsbelegung I alle Prämissen wahr sind,
dann ist auch die Konklusion wahr in I.“

$I(A)$	$I(B)$	A	$A \vee B$	$\models_I$	B
1	1	1	1	✓	1
1	0	1	1	↯	0
0	1	0	1	✓	1
0	0	0	0	✓	0

Logische Konsequenz

$F_1, \ldots, F_n \models G$: $F_1, \ldots, F_n \models_I G$ gilt für alle Interpretationen I.

„Die Formel G ist eine logische Konsequenz der Formeln $F_1, \ldots, F_n$.“

„Die Formel G folgt aus den Formeln $F_1, \ldots, F_n$.“

Konvention: „$\models G$“ ($n = 0$) bedeutet „G ist immer wahr (gültig).“

$A,\ A \vee B \models B$? Ist B logische Konsequenz von A und $A \vee B$? Nein!

$I(A)$	$I(B)$	$A,\ A \vee B \models_I B$			
1	1	1	1	✓	1
1	0	1	1	↯	0
0	1	0	1	✓	1
0	0	0	0	✓	0

$I(A) = 1$, $I(B) = 0$:
Es gilt $\mathrm{val}_I(A) = \mathrm{val}_I(A \vee B) = 1$,
aber $\mathrm{val}_I(B) \neq 1$!

I heißt Gegenbeispiel.

Man schreibt in diesem Fall auch $A,\ A \vee B \not\models B$.

$A,\ A \Rightarrow B \models B$? Ist B logische Konsequenz von A und $A \Rightarrow B$? Ja!

$I(A)$	$I(B)$	$A,\ A \Rightarrow B \models_I B$			
1	1	1	1	✓	1
1	0	1	0	✓	0
0	1	0	1	✓	1
0	0	0	1	✓	0

$$\frac{A \qquad A \Rightarrow B}{B} \qquad \frac{x \qquad \text{Wenn } x\text{, dann } y.}{y}$$

Ist eine gültige Inferenzregel!

Kriterium für die Gültigkeit von Inferenzregeln

Immer wenn alle Prämissen wahr sind, ist auch die Konklusion wahr.

Abgrenzung Konsequenz von Implikation

- Konsequenz $A \models B$:
  „Immer wenn A in einer Interpretation I wahr ist, ist auch B in I wahr (Interpretationen, in denen A falsch ist, werden nicht betrachtet).“

  Konsequenz ist eine semantische Aussage über die Wahrheit von Formeln.
- Implikation $A \Rightarrow B$:
  „Die Formel ist wahr, wenn A falsch oder B wahr ist (ansonsten ist sie falsch).“

  Implikation verbindet zwei Formeln syntaktisch zu einer größeren Formel.
- $A \Rightarrow B$ teilt mit, um welche Formel es sich handelt,
  $\models A \Rightarrow B$ behauptet außerdem, dass die Formel gültig ist.

Äquivalenz, Konsequenz und Gültigkeit

Mit der logischen Konsequenz können wir auch symbolisch ausdrücken, dass eine Formel gültig / erfüllbar / widerlegbar/ unerfüllbar ist:

$\models F$: F ist in allen Interpretationen wahr
F ist gültig

$\not\models F$: F ist **nicht** in allen Interpretationen wahr
F ist widerlegbar

$\models \neg F$: $\neg F$ ist in allen Interpretationen wahr
F ist in allen Interpretationen falsch
F ist unerfüllbar

$\not\models \neg F$: $\neg F$ ist **nicht** in allen Interpretationen wahr
F ist **nicht** in allen Interpretationen falsch
F ist erfüllbar

Äquivalenz, Konsequenz und Gültigkeit

Die Formeln F und G sind äquivalent ($F = G$) genau dann,
wenn $F \Leftrightarrow G$ eine gültige Formel ist.

Deduktionstheorem

G folgt aus $F_1, \ldots, F_n$ genau dann, wenn $F_n \Rightarrow G$ aus $F_1, \ldots, F_{n-1}$ folgt.

$F_1, \ldots, F_n \models G$ genau dann, wenn $F_1, \ldots, F_{n-1} \models F_n \Rightarrow G$.

Mehrfache Anwendung liefert:

$F_1, \ldots, F_n \models G$ genau dann, wenn $F_1 \Rightarrow (F_2 \Rightarrow \cdots (F_n \Rightarrow G) \cdots)$ gültig.

Wegen $A \Rightarrow (B \Rightarrow C) = (A \wedge B) \Rightarrow C$ erhalten wir weiters:

$F_1, \ldots, F_n \models G$ genau dann, wenn $(F_1 \wedge \cdots \wedge F_n) \Rightarrow G$ gültig.

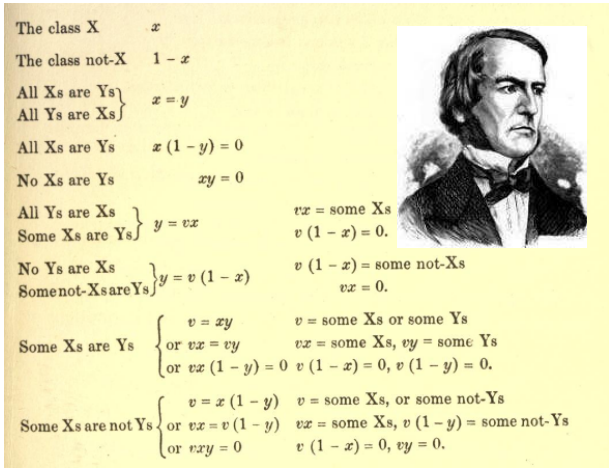
Das heißt: Semantik ($=$ und $\models$) ausdrückbar in der Syntax ($\Leftrightarrow$ und $\Rightarrow$).

Ist nicht in jeder Logik möglich!

Aussagenlogik – Übersicht

George Boole

- Der englische Mathematiker George Boole (1815–1864) begann, Logik und ihre Gesetze formal zu beschreiben
  - Entwickelte dazu 1847 die Algebra der Logik: „Boolesche Algebra“
- Diese arbeitet mit den Werten „falsche Aussage“ und „wahre Aussage“
- Abbildung auf 0 und 1



The class X	x	
The class not-X	$1 - x$	
All Xs are Ys All Ys are Xs	$x = y$	
All Xs are Ys	$x(1-y) = 0$	
No Xs are Ys	$xy = 0$	
All Ys are Xs Some Xs are Ys	$y = vx$	vx = some Xs $v(1-x) = 0$.
No Ys are Xs Some not-Xs are Ys	$y = v(1-x)$	$v(1-x)$ = some not-Xs $vx = 0$.
Some Xs are Ys	$v = xy$ or $vx = vy$ or $vx(1-y) = 0$	v = some Xs or some Ys vx = some Xs, vy = some Ys $v(1-x) = 0$, $v(1-y) = 0$.
Some Xs are not Ys	$v = x(1-y)$ or $vx = v(1-y)$ or $vxy = 0$	v = some Xs, or some not-Ys vx = some Xs, $v(1-y)$ = some not-Ys $v(1-x) = 0$, $vy = 0$.

Was ist eine Algebra? (1/2)

Algebraische Theorie (algebraische Spezifikation)

Operatorenliste + Axiome (Gleichungen über den Operatoren)

Beispiel: Algebraische Theorie der Gruppen

- Operatorenliste: $\circ/2$ (= zweistellig), $i/1$ (= einstellig), $e/0$ (= nullstellig, Konstante)
- Axiome:

(A)	$(x \circ y) \circ z = x \circ (y \circ z)$	Assoziativität von $\circ$
(N)	$e \circ x = x$	e ist links-neutrales Element
(K)	$i(x) \circ x = e$	$i(x)$ ist das Links-Komplement zu x

Was ist eine Algebra? (2/2)

Algebra zu einer algebraischen Theorie

- Grundmenge (Trägermenge)
- Funktionen auf der Grundmenge, für jeden Operator der Theorie eine
- Funktionen erfüllen die Axiome der algebraischen Theorie

Beispiel: $\langle \mathbb{Z},\ +,\ -,\ 0 \rangle$ ist eine Gruppe

$\mathbb{Z}$ ... Menge der ganzen Zahlen
$+$ ... Addition ganzer Zahlen, entspricht $\circ/2$
$-$ ... einstelliges Minus, entspricht $i/1$
0 ... Konstante 0, entspricht $e/0$
$\langle \mathbb{Z},\ +,\ -,\ 0 \rangle$ erfüllt die Gruppenaxiome, da gilt:

(A)	$(x + y) + z = x + (y + z)$	Assoziativität von $+$
(N)	$0 + x = x$	0 ist links-neutrales Element
(K)	$-x + x = 0$	$-x$ ist das Links-Komplement zu x

Was haben wir davon?

Was haben wir davon, wenn wir wissen,

- dass Addition, Komplement und 0 eine Gruppe bilden?
- dass bestimmte Funktionen eine algebraische Spezifikation erfüllen?

Alles, was wir in einer algebraischen Spezifikation ohne Bezug auf konkrete Funktionen herleiten können, gilt dann automatisch für jede Algebra $\Rightarrow$ Arbeitsersparnis

Wir müssen nur einmal zeigen, dass die Axiome der Spezifikation erfüllt sind.

Aus den Gleichungen (A), (N) und (K) folgt, dass e auch rechts-neutral und $i(x)$ auch das Rechts-Komplement ist:

$$x \circ e = x \quad \text{und} \quad x \circ i(x) = e$$

Gilt, obwohl $\circ$ nicht kommutativ ist!

Daher gelten diese abgeleiteten Eigenschaften auch für die ganzen Zahlen:

$$x + 0 = x \quad \text{und} \quad x + -x = 0$$

Boolesche Algebren (1/2)

Axiome der booleschen Algebren

- Operatoren: $\wedge/2$, $\vee/2$, $\neg/1$, $\bot/0$, $\top/0$
- Axiome:

Assoziativität:	$(x \wedge y) \wedge z = x \wedge (y \wedge z)$	$(x \vee y) \vee z = x \vee (y \vee z)$
Kommutativität:	$x \wedge y = y \wedge x$	$x \vee y = y \vee x$
Neutrale Elemente:	$x \wedge \top = x$	$x \vee \bot = x$
Komplement:	$x \wedge \neg x = \bot$	$x \vee \neg x = \top$
Distributivität:	$x \wedge (y \vee z) = (x \wedge y) \vee (x \wedge z)$	$x \vee (y \wedge z) = (x \vee y) \wedge (x \vee z)$

Die Aussagenlogik ist eine boolesche Algebra.

$\langle \mathbb{B}, \text{and}, \text{or}, \text{not}, 0, 1 \rangle$ erfüllt die Axiome.

Lässt sich beweisen, indem man jede der obigen Gleichung mit eine Wahrheitstabelle überprüft.

Boolesche Algebren (2/2)

Es gibt zahlreiche weitere boolesche Algebren.

Fuzzy Logik

Grundmenge: das reelle Intervall $[0, 1]$
$\wedge \ldots$ Minimum $\min(x, y)$
$\vee \ldots$ Maximum $\max(x, y)$
$\neg \ldots$ Komplement $1 - x$
$\bot \ldots 0$
$\top \ldots 1$

Mengen

Grundmenge: alle Teilmengen einer Menge M
$\wedge \ldots$ Durchschnitt $x \cap y$
$\vee \ldots$ Vereinigung $x \cup y$
$\neg \ldots$ Komplement bzgl. M, $M \setminus x$
$\bot \ldots$ leere Menge $\{\}$
$\top \ldots$ Gesamtmenge M

Die Aussagenlogik ist die *initiale* boolesche Algebra.
Das bedeutet, dass sie gewissermaßen die einfachste boolesche Algebra ist. Alles, was in der Aussagenlogik gilt, gilt auch in jeder anderen booleschen Algebra (aber nicht umgekehrt).

Eigenschaften boolescher Algebren

Aus den Axiomen der booleschen Algebra lassen sich zahlreiche weitere Eigenschaften und Gesetze ableiten. Eine Auswahl:

Vereinfachungsregeln

Idempotenz:	$x \wedge x = x$	$x \vee x = x$
Absorption:	$x \wedge (x \vee y) = x$	$x \vee (x \wedge y) = x$
Null-/Einsgesetz:	$x \wedge \bot = \bot$	$x \vee \top = \top$
Konsensgesetz:	$(x \wedge y) \vee (\neg x \wedge z) \vee (y \wedge z) = (x \wedge y) \vee (\neg x \wedge z)$	
	$(x \vee y) \wedge (\neg x \vee z) \wedge (y \vee z) = (x \vee y) \wedge (\neg x \vee z)$	

Verschieben und Elimination der Negation

de Morgan Regel:	$\neg(x \wedge y) = \neg x \vee \neg y$	$\neg(x \vee y) = \neg x \wedge \neg y$
Doppelnegation:	$\neg\neg x = x$	
Null-/Einskomplement:	$\neg\bot = \top$	$\neg\top = \bot$

Vereinfachte Klammerung

Wir haben die Syntax der Aussagenlogik folgendermaßen definiert.

Syntax aussagenlogischer Formeln

Die Menge $\mathcal{A}$ der aussagenlogischen Formeln ist die kleinste Menge, für die gilt:

(a1)–(a3) ...

(a4) $(F * G) \in \mathcal{A}$, wenn $F, G \in \mathcal{A}$ und $* \in \{\wedge, \uparrow, \vee, \downarrow, \Leftrightarrow, \not\Leftrightarrow, \Rightarrow, \Leftarrow\}$.

Binäre Operationen müssen immer geklammert werden, um die Auswertungsreihenfolge festzulegen. Da $\wedge$ und $\vee$ assoziativ sind, beeinflusst die Reihenfolge aber nicht das Ergebnis.

Schreibvereinfachung

- Bei geschachtelten Konjunktionen ($\wedge$-Ausdrücken) können die inneren Klammern entfallen.
- Bei geschachtelten Disjunktionen ($\vee$-Ausdrücken) können die inneren Klammern entfallen.
- Die Klammern ganz außen können weggelassen werden.

$$A \wedge B \wedge C \wedge D = (((A \wedge B) \wedge C) \wedge D) = ((A \wedge B) \wedge (C \wedge D)) = (A \wedge (B \wedge (C \wedge D)))$$

Die anderen Junktoren der Aussagenlogik

$$
\begin{aligned}
x \uparrow y &= \neg x \vee \neg y \\
x \downarrow y &= \neg x \wedge \neg y \\
x \Rightarrow y &= \neg x \vee y \\
x \Leftarrow y &= x \vee \neg y \\
x \Leftrightarrow y &= (\neg x \vee y) \wedge (x \vee \neg y) = (x \wedge y) \vee (\neg x \wedge \neg y) \\
x \not\Leftrightarrow y &= (\neg x \vee \neg y) \wedge (x \vee y) = (x \wedge \neg y) \vee (\neg x \wedge y)
\end{aligned}
$$

Mit all diesen Gesetzen lassen sich Formeln algebraisch vereinfachen.

$(A \uparrow B) \uparrow (\top \uparrow B)$	$x \uparrow y = \neg x \vee \neg y$
$= (\neg A \vee \neg B) \uparrow (\neg \top \vee \neg B)$	$x \uparrow y = \neg x \vee \neg y$
$= \neg(\neg A \vee \neg B) \vee \neg(\neg \top \vee \neg B)$	$\neg(x \vee y) = \neg x \wedge \neg y$
$= (\neg\neg A \wedge \neg\neg B) \vee (\neg\neg \top \wedge \neg\neg B)$	$\neg\neg x = x$
$= (A \wedge B) \vee (\top \wedge B)$	$x \wedge \top = x$, Kommutativität $\wedge$
$= (A \wedge B) \vee B$	$x \vee (x \wedge y) = x$ (Absorption), Kommutativität $\wedge, \vee$
$= B$	

Aussagenlogik – Übersicht

Bei der aussagenlogischen Modellierung gaben wir folgende Empfehlung.

Rezept für Zweifelsfälle der aussagenlogischen Modellierung

1. Identifiziere die elementaren Aussagen.
2. Analysiere *alle* Wahrheitsbelegungen.
3. Wähle geeignete logische Funktionen (unbeirrt von Intuition und natürlicher Sprache)

Klingt ja nicht schlecht, aber:

Wie kann man eine beliebige Funktion auf eine Kombination der bekannten logischen Grundfunktionen zurückführen?

Beziehungsweise:

Wie kann man eine beliebige Funktion mit den bekannten Operatoren als Formel darstellen?

Gesucht: Ein allgemeines Verfahren (ein Algorithmus), das zu einer gegebenen Funktion eine passende Formel liefert.

Von der Funktion zur Formel: Disjunktive Normalform

Gegeben: Funktion $f : \mathbb{B}^n \to \mathbb{B}$ (z.B. als Wahrheitstabelle)
Gesucht: Formel, die f darstellt

A	B	C	$F[A, B, C]$?
x	y	z	$f(x, y, z)$
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1

Von der Funktion zur Formel: Disjunktive Normalform

Gegeben: Funktion $f\colon \mathbb{B}^n \to \mathbb{B}$ (z.B. als Wahrheitstabelle)
Gesucht: Formel, die f darstellt

A	B	C	$F[A,B,C] := (\neg A \wedge B \wedge C) \vee (A \wedge \neg B \wedge \neg C) \vee (A \wedge B \wedge C)$			
x	y	z	$f(x,y,z)$			
0	0	0	0	0	0	0
0	0	1	0	0	0	0
0	1	0	0	0	0	0
0	1	1	1	1	0	0
1	0	0	1	0	1	0
1	0	1	0	0	0	0
1	1	0	0	0	0	0
1	1	1	1	0	0	1

$F[A,B,C]$ ... disjunktive Normalform zur Funktion f (DNF_f)

Von der Funktion zur Formel

Gegeben: Funktion $f\colon \mathbb{B}^n \to \mathbb{B}$ (z.B. als Wahrheitstabelle)
Gesucht: Formel, die f darstellt

A	B	C	$F[A,B,C]$?
x	y	z	$f(x,y,z)$
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1

Von der Funktion zur Formel

Gegeben: Funktion $f\colon \mathbb{B}^n \to \mathbb{B}$ (z.B. als Wahrheitstabelle)
Gesucht: Formel, die f darstellt

A	B	C	$F[A,B,C] := (A \vee B \vee C) \wedge (A \vee B \vee \neg C) \wedge (A \vee \neg B \vee C) \wedge (\neg A \vee B \vee \neg C) \wedge (\neg A \vee \neg B \vee C)$					
x	y	z	$f(x,y,z)$					
0	0	0	0	0	1	1	1	1
0	0	1	0	1	0	1	1	1
0	1	0	0	1	1	0	1	1
0	1	1	1	1	1	1	1	1
1	0	0	1	1	1	1	1	1
1	0	1	0	1	1	1	0	1
1	1	0	0	1	1	1	1	0
1	1	1	1	1	1	1	1	1

$F[A,B,C]$ ... konjunktive Normalform zur Funktion f (KNF_f)

Zusammenfassung

A_1	A_2	A_3	Konjunkte			Disjunkte		
0	0	0	0			$A_1 \vee$	$A_2 \vee$	$A_3 =: D_{000}$
0	0	1	0			$A_1 \vee$	$A_2 \vee \neg A_3 =: D_{001}$	
0	1	0	0			$A_1 \vee \neg A_2 \vee$		$A_3 =: D_{010}$
0	1	1	1	$\neg A_1 \wedge$	$A_2 \wedge$	$A_3 =: K_{011}$		
1	0	0	1	$A_1 \wedge \neg A_2 \wedge$		$\neg A_3 =: K_{100}$		
1	0	1	0			$\neg A_1 \vee$	$A_2 \vee \neg A_3 =: D_{101}$	
1	1	0	0			$\neg A_1 \vee \neg A_2 \vee$		$A_3 =: D_{110}$
1	1	1	1	$A_1 \wedge$	$A_2 \wedge$	$A_3 =: K_{111}$		

$\mathrm{DNF}_f = K_{011} \vee K_{100} \vee K_{111}$

$\mathrm{KNF}_f = D_{000} \wedge D_{001} \wedge D_{010} \wedge D_{101} \wedge D_{110}$

Folgerung:

$\{\text{not}, \text{and}, \text{or}\}$ ist funktional vollständig.

Aussagenlogik – Übersicht

Normalformen

Negationsnormalform

Normalform

Klasse (Menge) von Formeln, deren Struktur bestimmte syntaktische Einschränkungen erfüllt.

- Der eingeschränkte Aufbau erleichtert die (maschinelle) Verarbeitung der Formeln.
- Viele Verfahren/Tools akzeptieren als Eingabe nur Formeln in gewissen Normalformen.
- Die meisten Normalformen sind allgemein genug, um zu jeder Formel eine äquivalente Normalform finden zu können.

Negationsnormalform (NNF)

Eine Formel ist in Negationsnormalform (NNF), wenn

- nur die Operatoren $\wedge$, $\vee$ und $\neg$ vorkommen
- und Negationen nur unmittelbar vor Variablen auftreten.

$A \vee (\neg B \wedge (\neg A \vee C))$ ist in NNF.
$A \vee \neg(B \wedge (A \Rightarrow C))$ ist nicht in NNF.

Normalformen

Konjunktive Normalform

Literal: Unter Literalen verstehen wir

- Variablen x und
- negierte Variablen $\neg x$

Konjunktive Normalform (KNF)

Eine Formel ist in **konjunktiver Normalform**, wenn sie eine Konjunktion von Disjunktionen von Literalen ist. Sie besitzt also die Form

$$(L_{1,1} \vee \cdots \vee L_{1,m_1}) \wedge \cdots \wedge (L_{n,1} \vee \cdots \vee L_{n,m_n}) \qquad L_{i,j} \ \ldots \ \text{Literale}$$

Weiters sind $\bot$ und $\top$ Formeln in KNF.

Beispiele

- $(\neg A \vee B) \wedge (A \vee \neg C \vee D) \wedge \neg C$
- $(\neg A \vee B \vee \neg X \vee D) \wedge (\neg B \vee \neg Y)$

Normalformen

Disjunktive Normalform (DNF)

Eine Formel ist in **disjunktiver Normalform**, wenn sie eine Disjunktion von Konjunktionen von Literalen ist. Sie besitzt also die Form

$$(L_{1,1} \wedge \cdots \wedge L_{1,m_1}) \vee \cdots \vee (L_{n,1} \wedge \cdots \wedge L_{n,m_n}) \qquad L_{i,j} \ldots \text{ Literale}$$

Weiters sind $\bot$ und $\top$ Formeln in DNF.

Beispiele

- $(\neg A \wedge B \wedge C) \vee (A \wedge \neg C \wedge D)$ — DNF
- $(\neg A \vee B) \wedge (A \vee \neg C \vee D) \wedge \neg B$ — KNF
- $\neg A \wedge B \wedge C$ — DNF und KNF
- $\neg A \vee B \vee C$ — DNF und KNF
- $(\neg A \vee B \vee C) \wedge (A \vee (B \wedge D))$ — weder noch

Normalformen

Kanonische KNF/DNF

V ... endliche Menge von Variablen („die Variablen, die uns gerade interessieren“)

- *Volldisjunktion bzw. Maxterm über V:* Disjunktion von Literalen, wobei jede Variable aus V genau einmal negiert oder unnegiert vorkommt.
- *Vollkonjunktion bzw. Minterm:* Konjunktion von Literalen, wobei jede Variable aus V genau einmal negiert oder unnegiert vorkommt.

Kanonische konjunktive Normalform (KKNF) über V

Konjunktion von paarweise verschiedenen Volldisjunktionen über V

Kanonische disjunktive Normalform (KDNF) über V

Disjunktion von paarweise verschiedenen Vollkonjunktionen über V

Beispiele mit $V = \{A, B, C, D\}$

- $(A \vee \neg B \vee C \vee \neg D) \wedge (\neg A \vee \neg B \vee C \vee \neg D)$ KKNF über V
- $(A \wedge \neg B \wedge C \wedge \neg D) \vee (\neg A \wedge \neg B \wedge C \wedge \neg D)$ KDNF über V

Umwandlung einer Formel in KNF/DNF

Gegeben: Aussagenlogische Formel F
Gesucht: Äquivalente Formel in DNF/KNF

Semantische Methode:

1. Stelle die zu F gehörige Funktion f als Wahrheitstabelle dar.
2. Lies daraus DNF_f bzw. KNF_f ab.

Liefert eine KKNF bzw. KDNF über den Variablen von F

Algebraische Methode:

1. Ersetze alle Junktoren durch Ausdrücke mit $\wedge$, $\vee$ und $\neg$.
2. Verschiebe Negationen nach innen (de Morgan), eliminiere Doppelnegationen.
3. Wende das Distributivgesetz an.
   DNF: Schiebe Disjunktionen nach außen mittels $x \wedge (y \vee z) = (x \wedge y) \vee (x \wedge z)$
   KNF: Schiebe Konjunktionen nach außen mittels $x \vee (y \wedge z) = (x \vee y) \wedge (x \vee z)$
4. Vereinfache: eliminiere $\top$ und $\bot$ sowie komplementäre Literale.

Umwandlung einer Formel in KNF/DNF

Beispiel semantische Methode

Gegeben: $F = (A \Rightarrow (B \Leftrightarrow C)) \wedge (\neg A \Rightarrow (B \wedge C))$

A	B	C	F	Minterme	Maxterme
0	0	0	0		$D_{000} := A \vee B \vee C$
0	0	1	0		$D_{001} := A \vee B \vee \neg C$
0	1	0	0		$D_{010} := A \vee \neg B \vee C$
0	1	1	1	$K_{011} := \neg A \wedge B \wedge C$	
1	0	0	1	$K_{100} := A \wedge \neg B \wedge \neg C$	
1	0	1	0		$D_{101} := \neg A \vee B \vee \neg C$
1	1	0	0		$D_{110} := \neg A \vee \neg B \vee C$
1	1	1	1	$K_{111} := A \wedge B \wedge C$	

KDNF: $K_{111} \vee K_{100} \vee K_{011}$
KKNF: $D_{110} \wedge D_{101} \wedge D_{010} \wedge D_{001} \wedge D_{000}$

Umwandlung einer Formel in KNF/DNF

Beispiel algebraische Methode

Gegeben: $F = (A \Rightarrow (B \Leftrightarrow C)) \wedge (\neg A \Rightarrow (B \wedge C))$

1. Ersetze alle Junktoren durch Ausdrücke mit $\wedge$, $\vee$ und $\neg$.
   $(\neg A \vee (B \Leftrightarrow C)) \wedge (\neg\neg A \vee (B \wedge C))$ $\qquad x \Rightarrow y = \neg x \vee y$
   $(\neg A \vee (B \wedge C) \vee (\neg B \wedge \neg C)) \wedge (\neg\neg A \vee (B \wedge C))$ $\qquad x \Leftrightarrow y = (x \wedge y) \vee (\neg x \wedge \neg y)$
2. Verschiebe Negationen nach innen (de Morgan), eliminiere Doppelnegationen.
   $(\neg A \vee (B \wedge C) \vee (\neg B \wedge \neg C)) \wedge (A \vee (B \wedge C))$ $\qquad \neg\neg x = x$
3. Wende das Distributivgesetz an.
   DNF: Schiebe Disjunktionen nach außen mittels $x \wedge (y \vee z) = (x \wedge y) \vee (x \wedge z)$
   $(\neg A \wedge A) \vee (\neg A \wedge B \wedge C) \vee (B \wedge C \wedge A) \vee (B \wedge C) \vee (\neg B \wedge \neg C \wedge A) \vee (\neg B \wedge \neg C \wedge B \wedge C)$
   KNF: Schiebe Konjunktionen nach außen mittels $x \vee (y \wedge z) = (x \vee y) \wedge (x \vee z)$
   $(\neg A \vee B \vee \neg B) \wedge (\neg A \vee B \vee \neg C) \wedge (\neg A \vee C \vee \neg B) \wedge (\neg A \vee C \vee \neg C) \wedge (A \vee B) \wedge (A \vee C)$
4. Vereinfache: eliminiere $\top$ und $\bot$ sowie komplementäre Literale.
   DNF: $(B \wedge C) \vee (A \wedge \neg B \wedge \neg C)$
   KNF: $(\neg A \vee B \vee \neg C) \wedge (\neg A \vee \neg B \vee C) \wedge (A \vee B) \wedge (A \vee C)$

Welche Methode ist besser?

Gefühlt: Die semantische Methode ist übersichtlicher.

Theoretisch: Beide Methoden sind schlecht, denn beide sind im schlechtesten Fall exponentiell.

- Semantische Methode: Aufwand immer exponentiell in Variablenzahl! Wahrheitstabelle besitzt $2^{\text{Variablenzahl}}$ Zeilen.
- Algebraische Methode: Schritt 3 (Distributivgesetz) ist aufwändig, kann zu einer exponentiellen Verlängerung der Formel führen.

Praktisch:

- Semantische Methode nur brauchbar bei Formeln mit *sehr* wenigen Variablen. *Immer* exponentiell in Variablenzahl, liefert *immer* die kanonische DNF/KNF.
- Algebraische Methode teilweise auch für große Formeln brauchbar, insbesondere mit Computerunterstützung. Kann auch kleine DNFs/KNFs liefern.

Aussagenlogik – Übersicht

Das Erfüllbarkeitsproblem der Aussagenlogik

Erfüllbarkeitsproblem (Satisfiability, SAT)

Gegeben: aussagenlogische Formel F
Frage: Ist F erfüllbar, d.h., gibt es ein $I \in \mathcal{I}$, sodass $\text{val}_I(F) = 1$?

Effiziente Verfahren zur Lösung von SAT sind wichtig in der Praxis:

- Viele praktische Aufgaben lassen sich als Probleme der Aussagenlogik formulieren, wie z.B.
  - Verifikation von Hard- und Software
  - Planungsaufgaben, Logistik-Probleme
- Die meisten aussagenlogischen Fragen lassen sich zu einem (Un)Erfüllbarkeitsproblem umformulieren:

$$
\begin{aligned}
G \text{ gültig} &\iff \neg G \text{ unerfüllbar} \\
G \text{ widerlegbar} &\iff \neg G \text{ erfüllbar} \\
G = H &\iff G \not\Leftrightarrow H \text{ unerfüllbar} \\
F_1, \ldots, F_n \models G &\iff F_1 \wedge \cdots \wedge F_n \wedge \neg G \text{ unerfüllbar}
\end{aligned}
$$

Methoden zur Lösung von SAT

Wahrheitstabelle:

- Berechne den Formelwert der Reihe nach für jede Interpretation.
  Antwort „ja“, sobald man den Wert 1 erhält; „nein“, wenn immer 0.
- Unbrauchbar, da exponentiell: $2^{\text{Variablenzahl}}$ Interpretationen!

Umwandlung in DNF:

- Wandle F in eine disjunktive Normalform um.
  Antwort „nein“, wenn man $\bot$ erhält; „ja“ sonst.
- Unbrauchbar: F meistens in Fast-KNF. Distributivgesetz verlängert F exponentiell.

SAT-Solver: Programme, die SAT lösen.

- Verwenden fortgeschrittene algebraische/graphenorientierte/logische Methoden mit besonderen Datenstrukturen.
- Können SAT für Formeln mit Millionen von Variablen lösen.
- Stand der Technik bei der Verifikation von Prozessoren etc.
- Aber: Exponentielle Laufzeit für manche Formelarten!

Minisat: ein kleiner, effizienter SAT-Solver

- www.minisat.se: „a small, yet efficient, SAT solver“
- Eingabe: KNF Formel via Textdatei im DIMACS Format
- DIMACS Format:
  - Kopfzeile: `p cnf <v> <c>`
    <v> . . . Anzahl Variablen; <c> . . . Anzahl Klauseln
  - Alle anderen Zeilen: eine Disjunktion der KNF Formel
  - Variablen: $X_1, \ldots, X_v$ dargestellt durch Index
  - positiv: $i \ldots X_i$, negativ: $-i \ldots \neg X_i$
  - Disjunktion: $X_i \quad \ldots \quad X_k \quad 0$
- Beispiel: $(X_1 \vee X_2 \vee X_3) \wedge (\neg X_1 \vee \neg X_3)$

Input:

```
p cnf 3 2
 1  2 3 0
-1 -3 0
```

Output:

```
CPU time: 0.001s
SAT -1 2 -3
```

Interpretation: $I(X_1) = 0, I(X_2) = 1, I(X_3) = 0$ bzw. $\neg X_1 \wedge X_2 \wedge \neg X_3$

Aussagenlogik – Übersicht

House

Max wird mit hohem Fieber und ausgeprägten Gliederschmerzen in das Spital eingeliefert. Dr. House diskutiert die Diagnose mit einer Kollegin.
House: „Wenn der Patient Fieber hat, handelt es sich um Grippe oder Erkältung.“
Cameron: „Wenn er keine starken Gliederschmerzen hat, dann hat er auch keine Grippe.“
House: „Jedenfalls weisen hohes Fieber und starke Gliederschmerzen immer auf Grippe hin.“
Cameron: “Er hat sicher nicht beide Krankheiten gleichzeitig.“

Wie lautet die Diagnose?
Wie lässt sie sich mit Hilfe der Aussagenlogik finden und begründen?

House – Wahl der Aussagenvariablen

Aussagenvariablen können nur Aussagen repräsentieren, die einen Wahrheitswert besitzen.
Einzelne Haupt-, Zeit- oder Eigenschaftswörter sind keine Aussagen!

Falsch: $A =$ „krank“ oder $A =$ „Fieber“.

Möglich: $A =$ „Max ist krank“ oder $A =$ „Der Patient hat Fieber“.

Max wird mit hohem Fieber und ausgeprägten Gliederschmerzen in das Spital eingeliefert.

„Max wird mit ... eingeliefert“ $= A$?

„Max hat hohes Fieber“ $= A$,
„Max hat ausgeprägte Gliederschmerzen“ $= B$ und
„Max wird in das Spital eingeliefert“ $= C$?

„Max hat hohes Fieber“ = „Max hat Fieber“ $= A$ und
„Max hat ausgeprägte Gl.schmerzen“ = „Max hat Gl.schmerzen“ $= B$?

House – Wahl der Aussagenvariablen

Dr. House diskutiert die Diagnose mit einer Kollegin.

„Dr. House diskutiert ... mit einer Kollegin“ $= D$?

Wenn der Patient Fieber hat, handelt es sich um Grippe oder Erkältung.

„Der Patient hat Fieber“ $= E$?

„Der Patient hat Fieber“ = „Max hat Fieber“ $= A$?

Cameron: “Er hat sicher nicht beide Krankheiten gleichzeitig.“

„Cameron sagt, dass er nicht beide Krankheiten gleichzeitig hat.“ $= F$?

„Max kann nicht beide Krankheiten gleichzeitig haben.“ $= F$?

- *Elimination von Abkürzungen und Referenzen*
  „Er hat beide Krankheiten“ = „P. hat Grippe“ + „P. hat Erkältung“
- *Generalisierung*: Zusammenfassen von gleichartigen Aussagen
- *Abstraktion*: Weglassen von Details
- *Konzentration auf das Wesentliche*: Identifikation der relevanten Teilaussagen

Aber:

- Was zusammengefasst wurde, kann nicht mehr getrennt analysiert werden.
- Was weggelassen wurde, kann nicht für die Argumentation verwendet werden.
- Was nicht zusammenfasst wurde, aber zusammengehört, muss durch zusätzliche Formeln in Beziehung gesetzt werden.

Was kann man zusammenfassen? Was weglassen? Was ist wesentlich?

House – Wahl der Aussagenvariablen

Max wird mit hohem Fieber und ausgeprägten Gliederschmerzen in das Spital eingeliefert. Dr. House diskutiert die Diagnose mit einer Kollegin.
House: „Wenn der Patient Fieber hat, handelt es sich um Grippe oder Erkältung.“
Cameron: „Wenn er keine starken Gliederschmerzen hat, dann hat er auch keine Grippe.“
House: „Jedenfalls weisen hohes Fieber und starke Gliederschmerzen immer auf Grippe hin.“
Cameron: “Er hat sicher nicht beide Krankheiten gleichzeitig.“

F ... „Max/Patient hat (hohes) Fieber.“
S ... „Max/Patient hat starke/ausgeprägte Gliederschmerzen.“
G ... „Max/Patient hat eine Grippe.“
E ... „Max/Patient hat eine Erkältung.“

House – aussagenlogische Modellierung

F ... „Max/Patient hat (hohes) Fieber.“
S ... „Max/Patient hat starke/ausgeprägte Gliederschmerzen.“
G ... „Max/Patient hat eine Grippe.“
E ... „Max/Patient hat eine Erkältung.“

Max wird mit hohem Fieber und ausgeprägten Gliederschmerzen in das Spital eingeliefert. Dr. House diskutiert die Diagnose mit einer Kollegin.	$F_1 := F \wedge S$
House: „Wenn der Patient Fieber hat, handelt es sich um Grippe oder Erkältung.“	$F_2 := F \Rightarrow (G \vee E)$
Cameron: „Wenn er keine starken Gliederschmerzen hat, dann hat er auch keine Grippe.“	$F_3 := \neg S \Rightarrow \neg G$
House: „Jedenfalls weisen hohes Fieber und starke Gliederschmerzen immer auf Grippe hin.“	$F_4 := (F \wedge S) \Rightarrow G$
Cameron: “Er hat sicher nicht beide Krankheiten gleichzeitig.“	$F_5 := \neg(G \wedge E)$

House – Diagnose

Finde alle Interpretationen I, in denen alle Formeln wahr sind.

Methode 1: Wahrheitstabelle

Vereinfachung: Prüfe nur Interpretationen, in denen $F_1 = F \wedge S$ wahr ist.

F	S	G	E	F_1	$F \Rightarrow (G \vee E)$	$\neg S \Rightarrow \neg G$	$(F \wedge S) \Rightarrow G$	$\neg(G \wedge E)$
1	1	0	0	1	0	1	0	1
1	1	0	1	1	1	1	0	1
1	1	1	0	1	1	1	1	1
1	1	1	1	1	1	1	1	0

$I(G) = 1$, $I(E) = 0 \Longrightarrow$ Die Diagnose lautet auf „Grippe“.

Methode 2: Umwandlung in KNF, SAT-Solver aufrufen

$$\underbrace{F \wedge S}_{F_1} \wedge \underbrace{(\neg F \vee G \vee E)}_{F_2} \wedge \underbrace{(S \vee \neg G)}_{F_3} \wedge \underbrace{(\neg F \vee \neg S \vee G)}_{F_4} \wedge \underbrace{(\neg G \vee \neg E)}_{F_5}$$

SAT-Solver liefert „erfüllbar“ sowie die Interpretation I mit $I(F) = I(S) = I(G) = 1$ und $I(E) = 0$.

Weitere Lösungen durch Ausschluss der bereits gefundenen mit der zusätzlichen Formel $F_6 = \neg(F \wedge S \wedge G \wedge \neg E) = \neg F \vee \neg S \vee \neg G \vee E$

SAT-Solver liefert für $F_1 \wedge \cdots \wedge F_5 \wedge F_6$ das Ergebnis „unerfüllbar“, es gibt also keine weiteren Lösungen.

Methode 3: Umwandlung in DNF und Vereinfachung

$$\overbrace{F}^{F_1} \wedge \overbrace{S \wedge (\neg F \vee G \vee E)}^{F_2} \wedge \overbrace{(S \vee \neg G)}^{F_3} \wedge \overbrace{(\neg F \vee \neg S \vee G)}^{F_4} \wedge \overbrace{(\neg G \vee \neg E)}^{F_5}$$

$$(\underbrace{(F \wedge S \wedge \neg F)}_{=\bot} \vee (F \wedge S \wedge G) \vee (F \wedge S \wedge E)) \wedge \cdots$$

$$((F \wedge S \wedge G) \vee (F \wedge S \wedge E)) \wedge (S \vee \neg G) \wedge \cdots$$

$$((F \wedge S \wedge G) \vee (F \wedge S \wedge E) \vee \underbrace{(F \wedge S \wedge G \wedge \neg G)}_{=\bot} \vee \underbrace{(F \wedge S \wedge E \wedge \neg G)}_{\text{Absorption } F \wedge S \wedge E}) \wedge \cdots$$

$$((F \wedge S \wedge G) \vee (F \wedge S \wedge E)) \wedge (\neg F \vee \neg S \vee G) \wedge \cdots$$

$$((F \wedge S \wedge G \wedge G) \vee (F \wedge S \wedge E \wedge G)) \wedge \cdots$$

$$((F \wedge S \wedge G) \vee \underbrace{(F \wedge S \wedge E \wedge G)}_{\text{Absorption } F \wedge S \wedge G}) \wedge \cdots$$

$$(F \wedge S \wedge G) \wedge (\neg G \vee \neg E)$$

$$(\underbrace{(F \wedge S \wedge G \wedge \neg G)}_{=\bot} \vee (F \wedge S \wedge G \wedge \neg E)$$

$F \wedge S \wedge G \wedge \neg E$ DNF mit Lösung $I(F) = I(S) = I(G) = 1$ und $I(E) = 0$

Aussagenlogik – Übersicht

Gone Maggie gone

„The Simpsons“, Staffel 20, Folge 13
Homer will mit Maggie, dem Hund Knecht Ruprecht und einem Glas mit Giftpillen auf die andere Seite des Flusses.

The Simpsons – aussagenlogische Modellierung

If all you have is a hammer, everything looks like a nail.

M_i ... Maggie
K_i ... Knecht Ruprecht
G_i ... Gift
H_i ... Homer

... befindet sich zum Zeitpunkt i auf der anderen Seite des Flusses.

- Zum Zeitpunkt i befinden sich alle auf dieser Flussseite.

$\text{AlleHier}(i) := \neg M_i \wedge \neg K_i \wedge \neg G_i \wedge \neg H_i$

- Zum Zeitpunkt i befinden sind alle auf der anderen Flussseite.

$\text{AlleDort}(i) := M_i \wedge K_i \wedge G_i \wedge H_i$

- Wenn sich Maggie und KR oder Maggie und das Gift am selben Flussufer befinden, muss Homer bei Maggie sein.

$\text{Sicher}(i) := ((M_i \Leftrightarrow K_i) \vee (M_i \Leftrightarrow G_i)) \Rightarrow (M_i \Leftrightarrow H_i)$

MH_i ... Maggie
KH_i ... Knecht Ruprecht
GH_i ... Gift
} ... fährt mit Homer über den Fluss (zw. den Zeitpunkten $i-1$ und i).

HH_i ... Homer fährt alleine über den Fluss (zwischen $i-1$ und i).

- Genau eine Überfahrt zwischen den Zeitpunkten $i-1$ und i.

Überfahrt(i) :=

$$(MH_i \wedge \neg KH_i \wedge \neg GH_i \wedge \neg HH_i) \vee (\neg MH_i \wedge KH_i \wedge \neg GH_i \wedge \neg HH_i) \vee$$
$$(\neg MH_i \wedge \neg KH_i \wedge GH_i \wedge \neg HH_i) \vee (\neg MH_i \wedge \neg KH_i \wedge \neg GH_i \wedge HH_i)$$

- Definition der Überfahrten:

DefÜberfahrt(i) :=

$$\begin{aligned}
&(MH_i \Rightarrow ((M_{i-1} \not\Leftrightarrow M_i) \wedge (K_{i-1} \Leftrightarrow K_i) \wedge (G_{i-1} \Leftrightarrow G_i) \wedge (H_{i-1} \not\Leftrightarrow H_i) \wedge (H_i \Leftrightarrow M_i))) \\
\wedge\ &(KH_i \Rightarrow ((M_{i-1} \Leftrightarrow M_i) \wedge (K_{i-1} \not\Leftrightarrow K_i) \wedge (G_{i-1} \Leftrightarrow G_i) \wedge (H_{i-1} \not\Leftrightarrow H_i) \wedge (H_i \Leftrightarrow K_i))) \\
\wedge\ &(GH_i \Rightarrow ((M_{i-1} \Leftrightarrow M_i) \wedge (K_{i-1} \Leftrightarrow K_i) \wedge (G_{i-1} \not\Leftrightarrow G_i) \wedge (H_{i-1} \not\Leftrightarrow H_i) \wedge (H_i \Leftrightarrow G_i))) \\
\wedge\ &(HH_i \Rightarrow ((M_{i-1} \Leftrightarrow M_i) \wedge (K_{i-1} \Leftrightarrow K_i) \wedge (G_{i-1} \Leftrightarrow G_i) \wedge (H_{i-1} \not\Leftrightarrow H_i))
\end{aligned}$$

Gesamtformel: Nach n Überfahrten sollen alle auf der anderen Seite sein.

$$\text{Simpsons}(n) := \text{AlleHier}(0) \wedge \text{AlleDort}(n) \wedge \bigwedge_{i=0}^{n} \text{Sicher}(i) \wedge \bigwedge_{i=1}^{n} \text{Überfahrt}(i) \wedge \bigwedge_{i=1}^{n} \text{DefÜberfahrt}(i)$$

Methode zum Lösen des Rätsels:

1. Errate die benötigte Zahl n der Überfahrten.
2. Finde eine erfüllende Interpretation für die Formel Simpsons(n) (z.B. mit Hilfe eines SAT-Solvers).

Eine mögliche Lösung:
$n = 7$, $I(MH_1) = I(HH_2) = I(GH_3) = I(MH_4) = I(KH_5) = I(HH_6) = I(MH_7) = 1$

Gone Maggie gone (Fortsetzung)

„The Simpsons“, Staffel 20, Folge 13

Eigenschaften aussagenlogischer Modellierungen

Vorteile:

- *deklarativ-statisch, nicht prozedural-dynamisch*
  Welche Eigenschaften sollen gelten?
  Nicht: Welche Schritte sind für Lösung erforderlich?
- *modular*
  Neue Bedingungen werden durch zusätzliche Formeln berücksichtigt.

Nachteile:

- *Erraten von Parametern*
  n muss durch Probieren gefunden werden
- *Große Zahl an Variablen und Formeln*
  Dynamik muss durch indizierte Variablen simuliert werden.
- *Frame Problem*
  Bei jeder Aktion muss auch definiert werden, was sich nicht ändert.
- *unintuitiv*
  Bei der Modellierung von Abläufen denkt man an Zustände und Übergänge, nicht an statische Bedingungen.

Aussagenlogik – Übersicht

Repräsentation boolescher Funktionen

- Wie kann man boolesche Funktionen kompakt repräsentieren?
- Wie lassen sich diese Repräsentationen effizient handhaben?
  - Überprüfung von Eigenschaften: Erfüllbarkeit, Gültigkeit
  - „Rechnen“ mit Repräsentationen

- **Wahrheitstabellen**
  - „Semantische Methode“
  - Vorteil: intuitiv verständlich, helfen beim Erlernen des Gebiets
  - Nachteile: Für n Variablen besitzt die Wahrheitstabelle 2^n Zeilen
    - Wir benötigen *immer* 2^n viele Zeilen
    - Nur verwendbar für sehr kleine Werte von n, in der Praxis irrelevant
- **Formeln** und andere termbasierte Darstellungen
  - „Algebraische Methoden“ (regelbasierte Transformationenen)
    - Beispielweise Umwandlung in Normalform mittels Algorithmus
  - Vorteile:
    - Darstellung kompakt, insbesondere bei Verwendung von Directed Acyclic Graphs (DAGs)
    - dadurch sind auch viele Operationen damit effizient
  - Nachteile:
    - intuitiv schwerer verständlich
    - verhältnismäßig komplexe Datenstruktur und Operationen
    - Regeln wie das Distributivgesetz können die Formelgröße verdoppeln.

Repräsentation Boolescher Funktionen

- **If-then-else (ITE)-Ausdrücke:** siehe unten
- **Binäre Entscheidungsdiagramme** (Binary Decision Diagrams, BDDs): siehe unten
- **Weitere Methoden:**
  - KV-Diagramme (Karnaugh-Veitch-Diagramme, Karnaugh map):
    - Methode zum Entwurf kleiner logischer Schaltkreise
    - Nur für sehr kleine Variablenzahlen (Lehrzwecke), in der Praxis irrelevant
  - Quine-McCluskey Algorithmus:
    - Minimierung von KNFs und DNFs
    - Brauchbar für praktisch relevante Probleme, mittlerweile veraltet
  - Graphbasierte Methoden unter Verwendung von Heuristiken (SAT-Solver, Espresso, ...)
    - Stand der Technik im Schaltungsentwurf und zur Verifikation von Hard- und Software

- Wahl der internen Datenstruktur entscheidend
  - Lineare Darstellungen (Text, Bäume) exponentiell schlechter als DAGs („structure sharing“)
    Algorithmen können eine schlechte Repräsentation nicht wett machen.
- Wahl der Darstellung und der Methoden abhängig von der geplanten Anwendung
  - Aus prinzipiellen Gründen gibt es keine Methode, die alle Anwendungen effizient behandelt (außer vielleicht, wenn „NP=P“)

ITE-Darstellung boolescher Funktionen

ITE ... dreistellige Operation, analog zu `if`-Ausdrücken (nicht `if`-Anweisungen!) in Programmiersprachen

x	y	z	ITE(x,y,z)
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	1

$$\text{ITE}(x,y,z) \hat{=} (x \wedge y) \vee (\neg x \wedge z)$$
$$\hat{=} (x \Rightarrow y) \wedge (\neg x \Rightarrow z)$$

JAVA:
Bedingung ? true-Wert : false-Wert

Pseudocode:

```
if x then
    y
else
    z
```

ITE-Form einer aussagenlogischen Funktion bzgl. der Variablen V

- 0 und 1 sind ITE-Formen.
- Literale über den Variablen V (x und $\neg x$ für $x \in V$) sind ITE-Formen.
- ITE(x, I_1, I_2) ist eine ITE-Form bzgl. V,
  wenn $x \in V$ ist und I_1, I_2 ITE-Formen bzgl. $V \setminus \{x\}$ sind.

Wahrheitstabelle und ITE-Form

x	y	z	f
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	1

```
if x then
    if y then
        if z then 1
        else 0
    else
        if z then 1
        else 0
else
    if y then
        if z then 1
        else 1
    else
        if z then 0
        else 0
```

$$\mathrm{ITE}(x, \mathrm{ITE}(y, \mathrm{ITE}(z, 1, 0), \mathrm{ITE}(z, 1, 0)), \mathrm{ITE}(y, \mathrm{ITE}(z, 1, 1), \mathrm{ITE}(z, 0, 0)))$$

Vereinfachte ITE-Formen

- ITE-Form abgelesen aus einer Wahrheitstabelle haben folgende Eigenschaften:
  - Die Variablen sind geordnet.
  - Die Länge der ITE-Form ist exponentiell in der Zahl der Variablen (wie die Wahrheitstabelle).
- Wenn wir in ITE-Formen auch Literale verwenden, können wir sie oft mit folgenden Gleichungen vereinfachen.

$$\mathrm{ITE}(x, 1, 0) = x$$
$$\mathrm{ITE}(x, 0, 1) = \neg x$$
$$\mathrm{ITE}(x, l, l) = l$$

Die ITE-Form von vorhin lässt sich auf diese Weise vereinfachen.

$$\begin{aligned}
&\mathrm{ITE}(x, \mathrm{ITE}(y, \underbrace{\mathrm{ITE}(z, 1, 0)}_{z}, \underbrace{\mathrm{ITE}(z, 1, 0)}_{z}), \mathrm{ITE}(y, \underbrace{\mathrm{ITE}(z, 1, 1)}_{1}, \underbrace{\mathrm{ITE}(z, 0, 0)}_{0})) \\
&= \mathrm{ITE}(x, \underbrace{\mathrm{ITE}(y, z, z)}_{z}, \underbrace{\mathrm{ITE}(y, 1, 0)}_{y}) \\
&= \mathrm{ITE}(x, z, y)
\end{aligned}$$

Formel und ITE-Form

- Logische Operatoren lassen sich als ITE-Form darstellen.

$$
\begin{aligned}
x &\rightarrow \mathrm{ITE}(x, 1, 0) \\
\neg x &\rightarrow \mathrm{ITE}(x, 0, 1) \\
x \wedge y &\rightarrow \mathrm{ITE}(x, y, 0) \\
x \vee y &\rightarrow \mathrm{ITE}(x, 1, y)
\end{aligned}
$$

- Ausgehend von einer DNF/KNF kann man nun alle Operatoren ersetzen.
- Andere Methode: Shannon-Zerlegung

Shannon-Zerlegung

Gegeben: Formel F

1. Wähle eine Variable x in F
2. Bestimme den *Kofaktor von F nach x und $\top$*:
   Wir ersetzen alle x in F durch $\top$ und nennen die Formel $F_{x,\top}$
3. Bestimme den *Kofaktor von F nach x und $\bot$*:
   Wir ersetzen alle x in F durch $\bot$ und nennen die Formel $F_{x,\bot}$
4. Dann gilt:

$$\begin{aligned} F &= (x \wedge F_{x,\top}) \vee (\neg x \wedge F_{x,\bot}) && \text{Shannon-Zerlegung} \\ &= (x \Rightarrow F_{x,\top}) \wedge (\neg x \Rightarrow F_{x,\bot}) && \text{duale Shannon-Zerlegung} \end{aligned}$$

- Die Kofaktoren $F_{x,\top}$ und $F_{x,\bot}$
  - lassen sich in der Regel vereinfachen
  - enthalten eine Variable weniger (x kommt nicht mehr vor)
  - können nun ihrerseits mit der Shannonzerlegung umgeformt werden
- Die Shannon-Zerlegung einer Formel entspricht einer Fallunterscheidung:
  - Falls x wahr ist, entspricht F der Formel $F_{x,\top}$
  - Falls x falsch ist, entspricht F der Formel $F_{x,\bot}$

Beispiel zur Shannon-Zerlegung 1

Gegeben: $F = (x \Rightarrow (y \wedge z)) \wedge (\neg y \uparrow (x \vee \neg z))$

1. Wähle z.B. die Variable x, dann gilt:

$$\begin{aligned} F_{x,\top} &= (\top \Rightarrow (y \wedge z)) \wedge (\neg y \uparrow (\top \vee \neg z)) \\ &= (y \wedge z) \wedge (\neg y \uparrow \top) \\ &= (y \wedge z) \wedge y \\ &= y \wedge z \\ F_{x,\bot} &= (\bot \Rightarrow (y \wedge z)) \wedge (\neg y \uparrow (\bot \vee \neg z)) \\ &= \top \wedge (\neg y \uparrow \neg z) \\ &= \neg y \uparrow \neg z \\ &= y \vee z \end{aligned}$$

2. Für F erhalten wir daher:

$$\begin{aligned} F &= (x \wedge F_{x,\top}) \vee (\neg x \wedge F_{x,\bot}) \\ &= (x \wedge y \wedge z) \vee (\neg x \wedge (y \vee z)) \end{aligned}$$

Beispiel zur Shannon-Zerlegung 2

Gegeben: $F = (x \Rightarrow (y \wedge z)) \wedge (\neg y \uparrow (x \vee \neg z))$

1. Wähle z.B. die Variable x, dann gilt (wie zuvor):

$$F_{x,\top} = (\top \Rightarrow (y \wedge z)) \wedge (\neg y \uparrow (\top \vee \neg z)) = y \wedge z$$
$$F_{x,\bot} = (\bot \Rightarrow (y \wedge z)) \wedge (\neg y \uparrow (\bot \vee \neg z)) = \neg y \uparrow \neg z$$

2. Als nächstes wählen wir die Variable y und zerlegen $G = F_{x,\top}$ und $H = F_{x,\bot}$.

$$G_{y,\top} = \top \wedge z = z \qquad H_{y,\top} = \neg\top \uparrow \neg z = \top$$
$$G_{y,\bot} = \bot \wedge z = \bot \qquad H_{y,\bot} = \neg\bot \uparrow \neg z = z$$

3. Für F erhalten wir daher:

$$\begin{aligned} F &= (x \wedge F_{x,\top}) \vee (\neg x \wedge F_{x,\bot}) \\ &= (x \wedge ((y \wedge G_{y,\top}) \vee (\neg y \wedge G_{y,\bot}))) \vee (\neg x \wedge ((y \wedge H_{y,\top}) \vee (\neg y \wedge H_{y,\bot}))) \\ &= (x \wedge ((y \wedge z) \vee (\neg y \wedge \bot))) \vee (\neg x \wedge ((y \wedge \top) \vee (\neg y \wedge z))) \\ &= (x \wedge y \wedge z) \vee (\neg x \wedge (y \vee (\neg y \wedge z))) \\ &= (x \wedge y \wedge z) \vee (\neg x \wedge (y \vee z)) \end{aligned}$$

ITE-Form zu einer Formel mittels Shannon-Zerlegung

Gesucht: ITE-Form zu einer Formel F

1. Wähle eine Variable in F. Sei x diese Variable.
2. Bestimme die ITE-Form zur Formel $F_{x,\top}$. Wir nennen sie $I_{x,\top}$.
3. Bestimme die ITE-Form zur Formel $F_{x,\bot}$. Wir nennen sie $I_{x,\bot}$.
4. Dann ist $\mathrm{ITE}(x, I_{x,\top}, I_{x,\bot})$ eine ITE-Form zur Formel F.

Fortsetzung Beispiel: ITE-Form zur vorigen Formel

$$G_{y,\top} = z \to \mathrm{ITE}(z, 1, 0)$$
$$G_{y,\bot} = \bot \to \mathrm{ITE}(z, 0, 0)$$
$$F_{x,\top} = (y \wedge G_{y,\top}) \vee (\neg y \wedge G_{y,\bot}) \to \mathrm{ITE}(y, \mathrm{ITE}(z, 1, 0), \mathrm{ITE}(z, 0, 0))$$

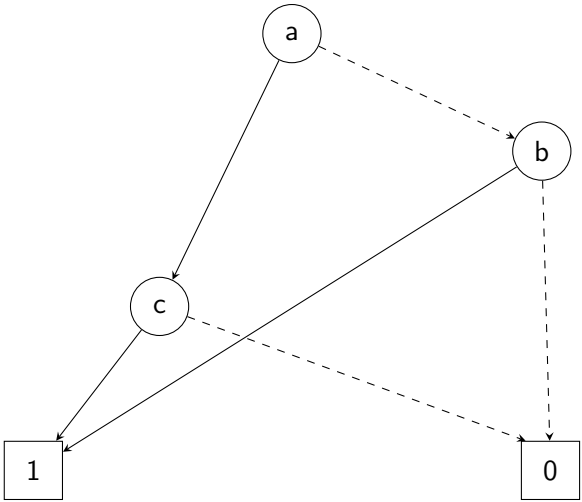
$$H_{y,\top} = \top \to \mathrm{ITE}(z, 1, 1)$$
$$H_{y,\bot} = z \to \mathrm{ITE}(z, 1, 0)$$
$$F_{x,\bot} = (y \wedge H_{y,\top}) \vee (\neg y \wedge H_{y,\bot}) \to \mathrm{ITE}(y, \mathrm{ITE}(z, 1, 1), \mathrm{ITE}(z, 1, 0))$$

$$F = (x \wedge F_{x,\top}) \vee (\neg x \wedge F_{x,\bot}) \to \mathrm{ITE}(x, \mathrm{ITE}(y, \mathrm{ITE}(z, 1, 0), \mathrm{ITE}(z, 0, 0)), \mathrm{ITE}(y, \mathrm{ITE}(z, 1, 1), \mathrm{ITE}(z, 1, 0)) \ )$$

Binäre Entscheidungsdiagramme / Binary Decision Diagrams (BDD)

- Binäre Entscheidungsdiagramme sind eine weitere Darstellung von Booleschen Funktionen
  - Englisch: Binary Decision Diagrams (BDD)
  - Boolesche Funktionen werden als Flussdiagramm dargestellt, das Auswertung der Funktion erlaubt
  - Führt teilweise zu schnellerer Auswertung von Funktionen
  - Kann auch als Form der Komprimierung der Booleschen Funktion angesehen sehen

BDD – Beispiel



$f = (a \wedge c) \vee (\neg a \wedge b)$

Negation auf der Kante

——→ nicht negiert

- - - -→ negiert

Pro „Ebene“ eine Variable
(hier zufällig: pro Ebene ein Knoten)

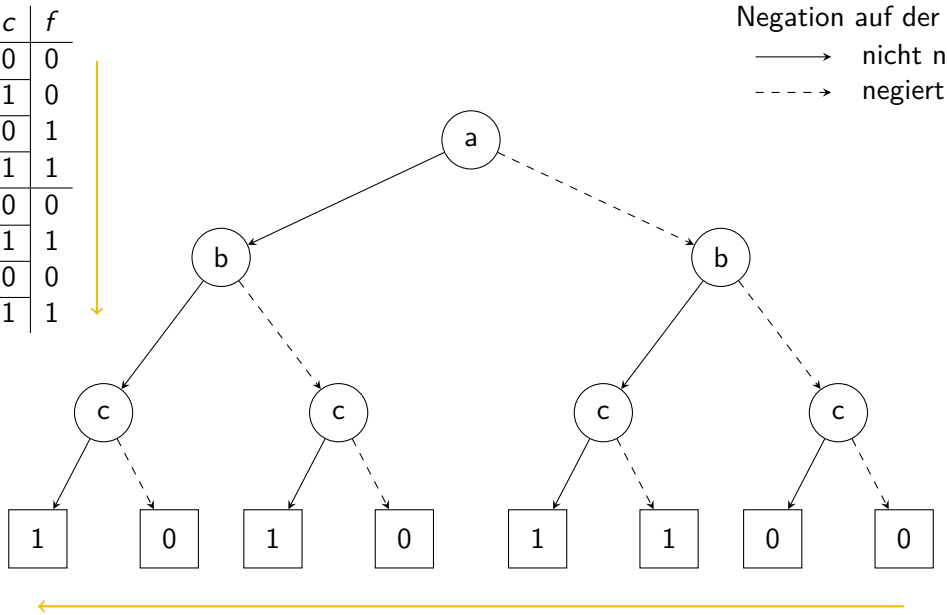
BDD – Erstellung

Grundsätzlich zwei Möglichkeiten BDDs zu erstellen:

- Ausgehend von Entscheidungsbaum (Decision Tree) + Vereinfachung
- Aus Wahrheitstabelle mit sog. Beads (basiert auf Shannon-Zerlegung)

Decision Tree

a	b	c	f
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	1

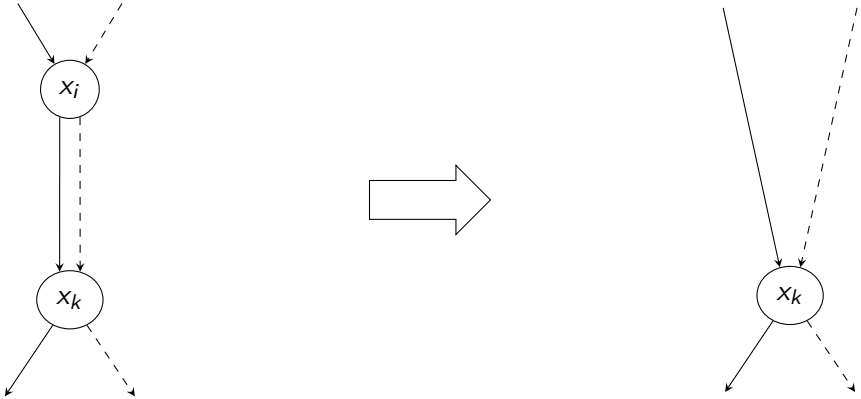


BDD - Reduction Rules

- Umwandlung eines Decision Trees in einen minimalen DAG (Directed Acyclic Graph)
- Nur ein 0 und ein 1 Knoten, danach wiederholtes Anwenden von zwei Regeln bis nicht mehr möglich:
  - **Delete Rule**: Löschen unnötiger Zwischenstufen
  - **Merge Rule**: Zusammenfassen von identen Teilgraphen
- Ergebnis: Binary Decision Diagram (BDD)
  - Eigentlich geordnetes BDD, engl. Ordered BDD = OBDD

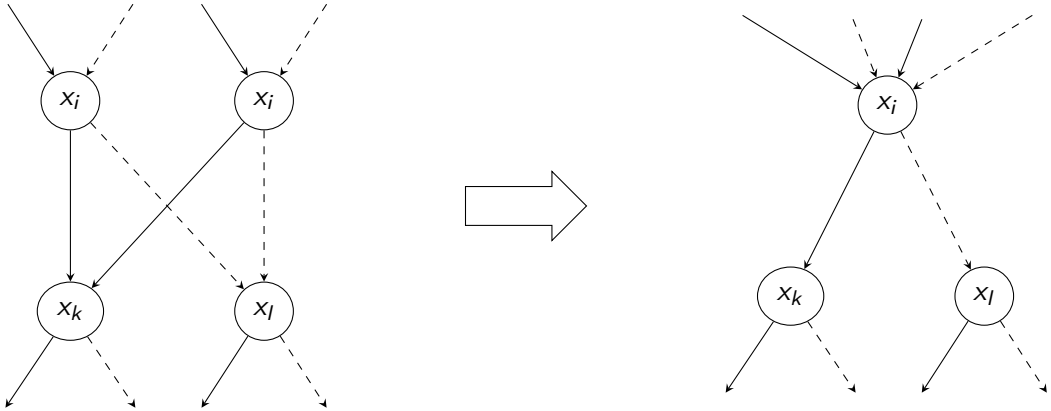
BDD - Reduction Rules

Delete Rule: Wenn beide ausgehende Kanten von x_i auf x_k zeigen, können wir x_i löschen und die eingehenden Kanten von x_i entsprechend auf x_k umlegen



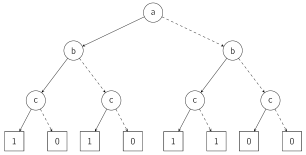
BDD - Reduction Rules

Merge Rule: Wenn wir zwei Kopien von x_i haben, die genau gleich auf x_k und x_l verweisen, können wir die Kopien von x_i mergen (zusammenlegen)

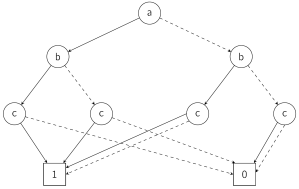


Beispiel

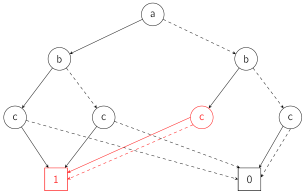
Decision Tree



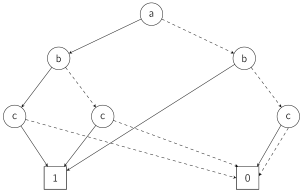
1: Nur ein 0 und 1 Knoten



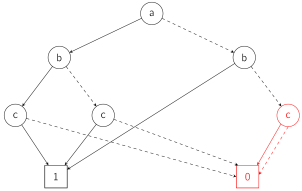
2: Delete Rule



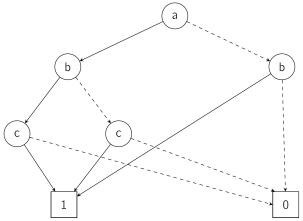
2: Delete Rule



3: Delete Rule

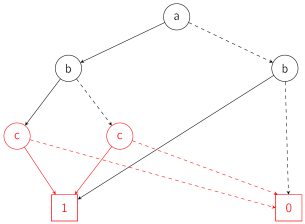


3: Delete Rule

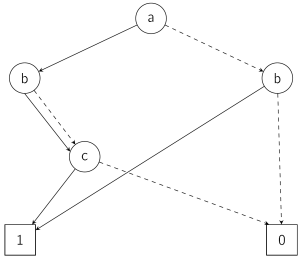


Beispiel

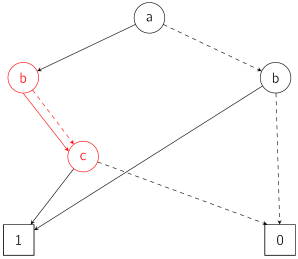
4: Merge rule



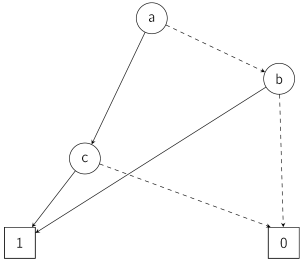
4: Merge rule



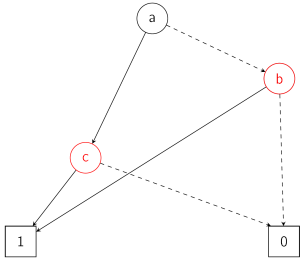
5: Delete rule



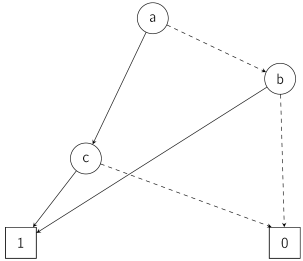
5: Delete rule



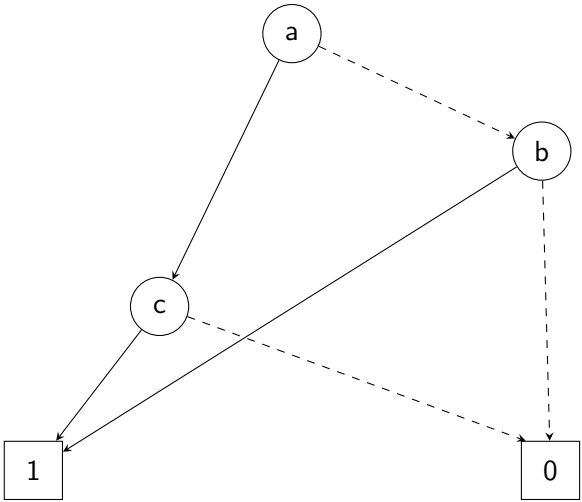
6: Merge rule?



6: Nein, weil $b \neq c$



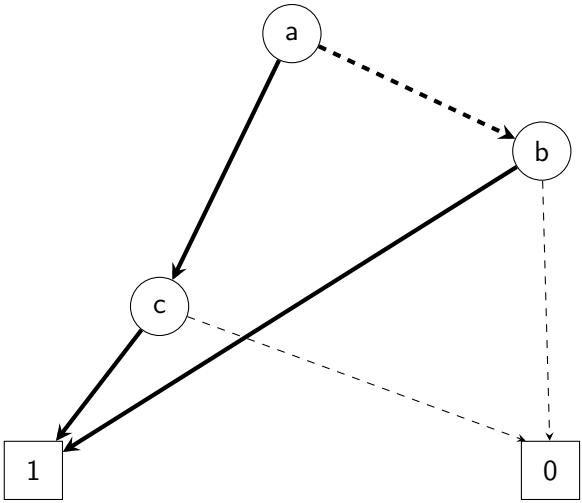
Beispiel – ITE-Form




```
if a then
    if c then 1
    else 0
else
    if b then 1
    else 0
```

$\mathsf{ITE}(a, \mathsf{ITE}(c, 1, 0), \mathsf{ITE}(b, 1, 0))$

Beispiel – Disjunktive Normalform



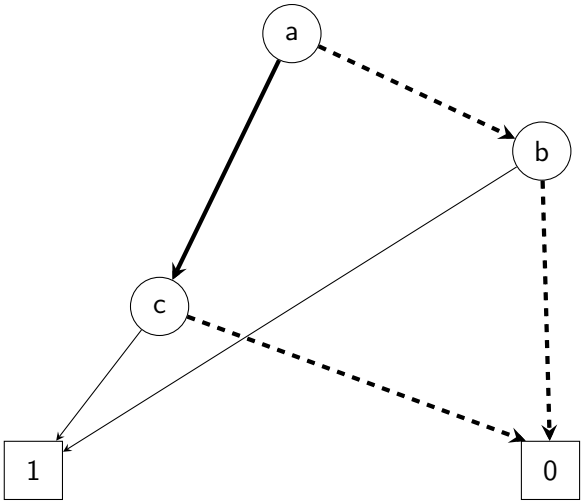
Disjunktive Normalform ablesen:

Pfade von Wurzel zu „1“

$(a \wedge c) \vee (\neg a \wedge b)$

Nicht notwendigerweise minimal!

Beispiel – Konjunktive Normalform



Konjunktive Normalform ablesen:

Pfade von Wurzel zu „0“
Variablen negieren

$(\neg a \vee c) \wedge (a \vee b)$

Nicht notwendigerweise minimal!

Vereinfachte Reduktion (Beads)

- Bisherige Reduktion im Baum durch Anwendung von
  - Merge Rule
  - Delete Rule
- Vereinfachte Version durch „Aufschreiben“ der Ergebnisse der Wahrheitstabelle als Wort der Länge 2^n, wobei n die Anzahl der Eingangsvariablen ist
  - Beispiel: $f(a, b, c) = 01011100$
- Sogenannte **Beads** (binäre Wörter der Länge 2^n)
  - **Squares** (Quadrate): Binäre Wörter der Form **ww**, wobei **w** ein Wort der Länge 2^{n-1} ist
  - **Beads**: binäre Wörter der Länge 2^n die keine Squares sind
  - Beispiele: 0000 0001 ist ein Bead, 1001 1001 ist ein Square
- Beispiel für $n = 2$:
  - Alle Beads der Länge 4 sind:
    0001, 0010, 0011, 0100, 0110, 0111,1000, 1001, 1011, 1100, 1101, 1110
  - Keine Beads der Länge 4 sind:
    0000, 0101, 1010, 1111

Beispiel mit Beads

Beispiel von Folie 95:

a	b	c	y
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	1

Auslesen des binären Wortes
1010 1100 der Länge 2^3

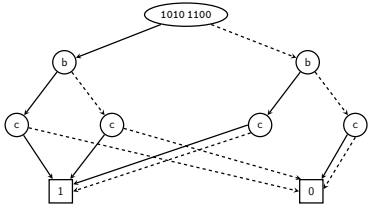
Beobachtung:
linke Hälfte entspricht $a = 1$,
rechte Hälfte entspricht $a = 0$.

Nachfolgend zwei Beispiele:

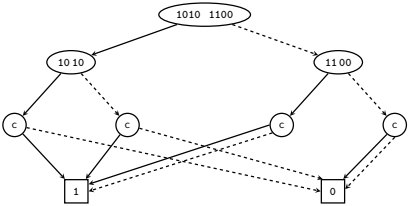
- Entscheidungsbaum vorhanden
- Ohne Entscheidungsbaum

Beispiel mit Beads

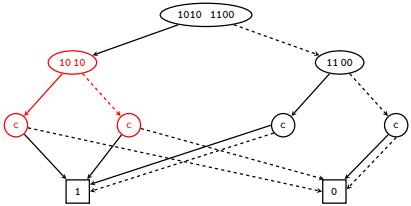
Wahrheitswerte in Wurzel eintragen



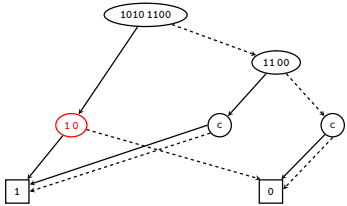
Halbieren und nach unten übertragen



Falls gleich **ww** . . .

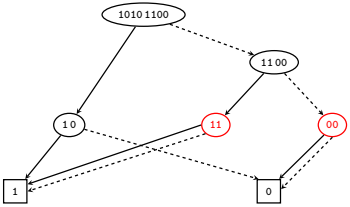


. . . Merge & Delete Rule

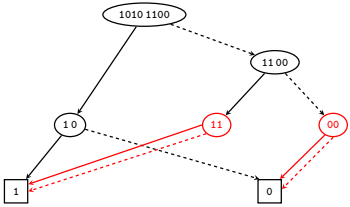


Beispiel mit Beads

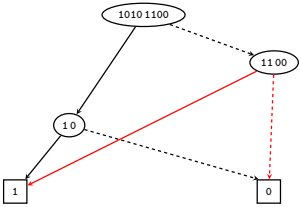
Halbieren und nach unten übertragen



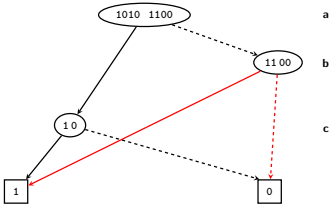
Falls gleich 00 . . . 0 oder 11 . . . 1



. . . direkt auf 0 oder 1 umleiten

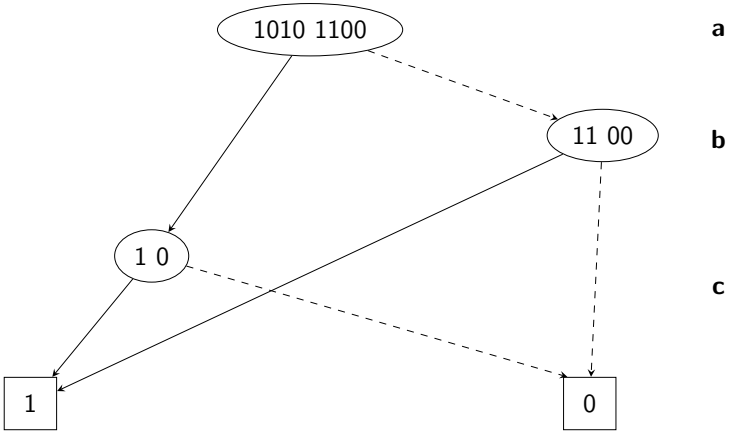


Variablen beschriften



Beispiel mit Beads ohne Decision Tree

Beispiel von Folie 95:
Wahrheitswerte eintragen; Halbieren und nach unten übertragen; Squares überspringen Ebenen, bis Beads übrig bleiben; Falls gleich $00\ldots0$ oder $11\ldots1$, je nach eingehender Kante direkt auf 0 oder 1 umleiten

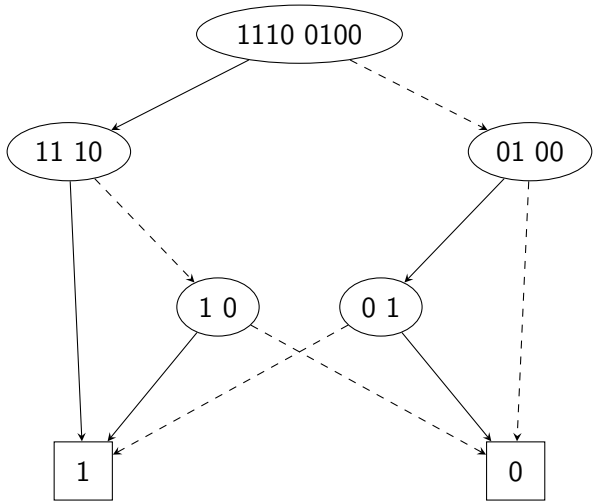


Variablenreihenfolge

Selbes Beispiel mit anderer Variablenreihenfolge:

c	b	a	y
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	1

Benötigt mehr Knoten!



Variablenreihenfolge

- Größe von BDDs hängt von Variablenreihenfolge in der Wahrheitstabelle ab
- Für n Variablen gibt es $n!$ unterschiedliche Permutationen
  - Für $n = 10$: $n! > 3.6 \cdot 10^6$
  - Für $n = 20$: $n! > 2.4 \cdot 10^{18}$

  $\Rightarrow$ Wächst exponentiell in der Anzahl der Variablen
- Minimale Form zu finden ist schwer

Größe von BDDs

- Es gibt Boolesche Funktionen, sodass unabhängig von der Variablenreihenfolge alle BDDs exponentielle Größe haben
- Die meisten praktisch relevanten Funktionen haben geringe Größe
- Bekannte schwer zu vereinfachende Funktion: *Multiplikation*
  - Unabhängig von der Variablenordnung exponentielle Größe der Vereinfachung
- Für manche exponentiell große disjunktive Normalform gibt es BDDs in polynomieller Größe: z.B.: *Majority-Funktion*
  - Funktion in n Variablen
  - Liefert false, wenn mehr als die Hälfte der Variablen den Wert false haben, ansonsten true

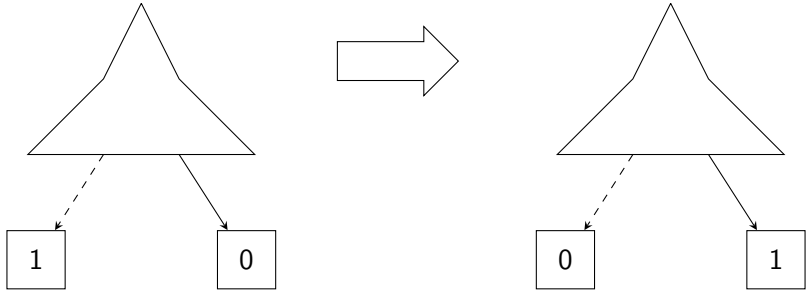
Eigenschaften von BDDs

- In einem BDD kommen **alle** „Beads der Wahrheitstabelle“ und **nur diese** vor (keine Squares)
- Darstellung einer bestimmten Booleschen Funktion ist bei gegebener Variablenreihenfolge **isomorph**
- Zwei isomorphe BDDs beschreiben äquivalente Boolesche Ausdrücke
- Typische Funktionen sind einfach mittels BDD-Darstellung zu realisieren
- Operationen können direkt auf der kompakten Darstellung durchgeführt werden

Typische Funktionen auf BDDs

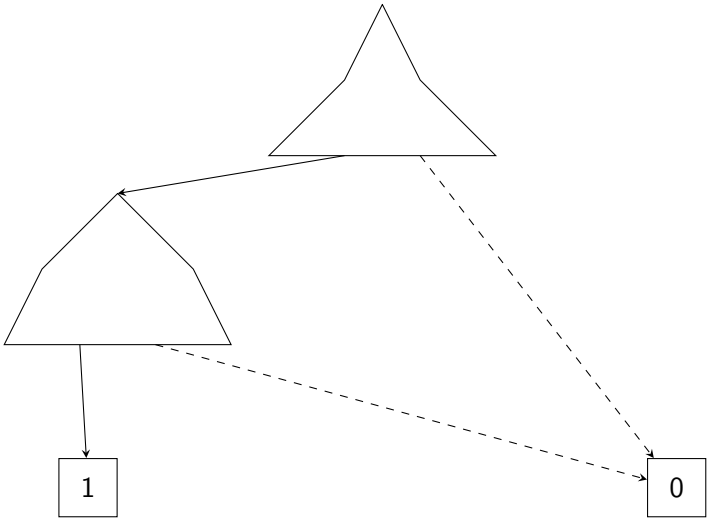
Negation

1 und 0 vertauschen



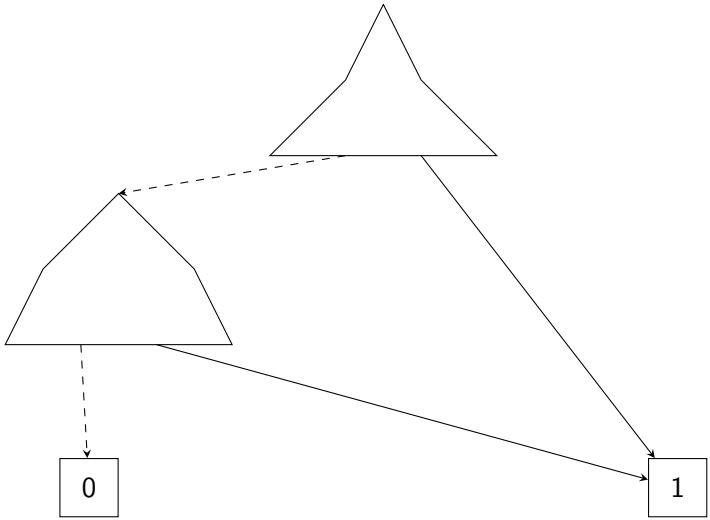
Typische Funktionen auf BDDs

UND



Typische Funktionen auf BDDs

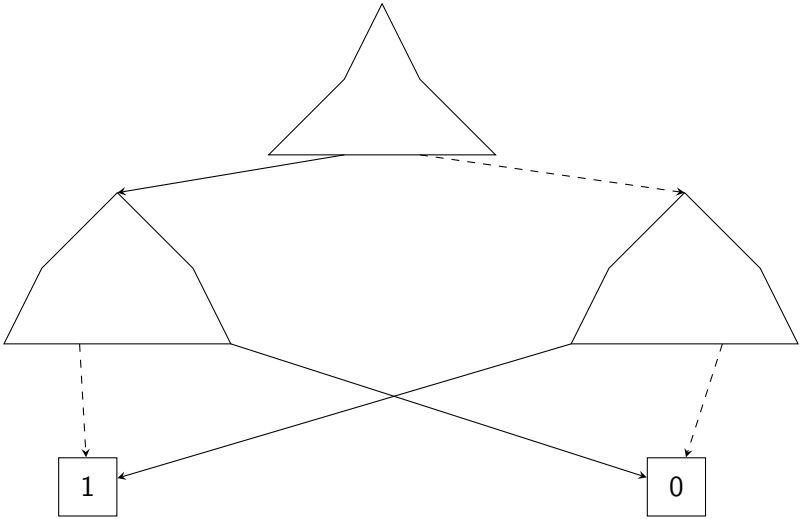
ODER



Typische Funktionen auf BDDs

XOR

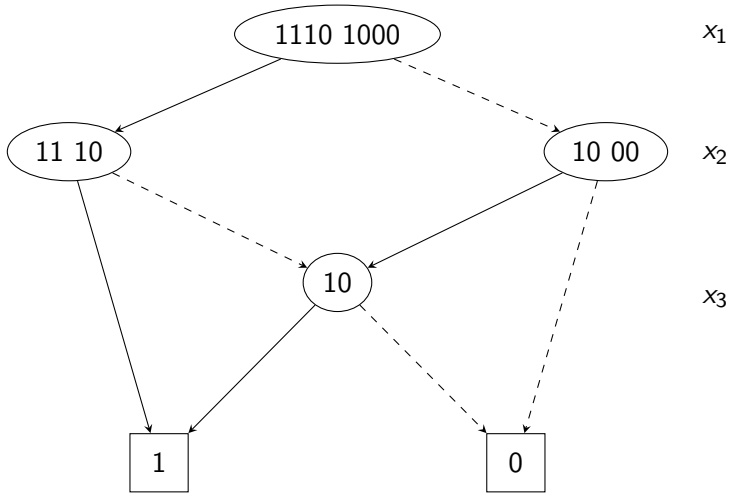
einen Operanden duplizieren



Beispiel

Majority-Funktion in drei Variablen

x_1	x_2	x_3	y
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

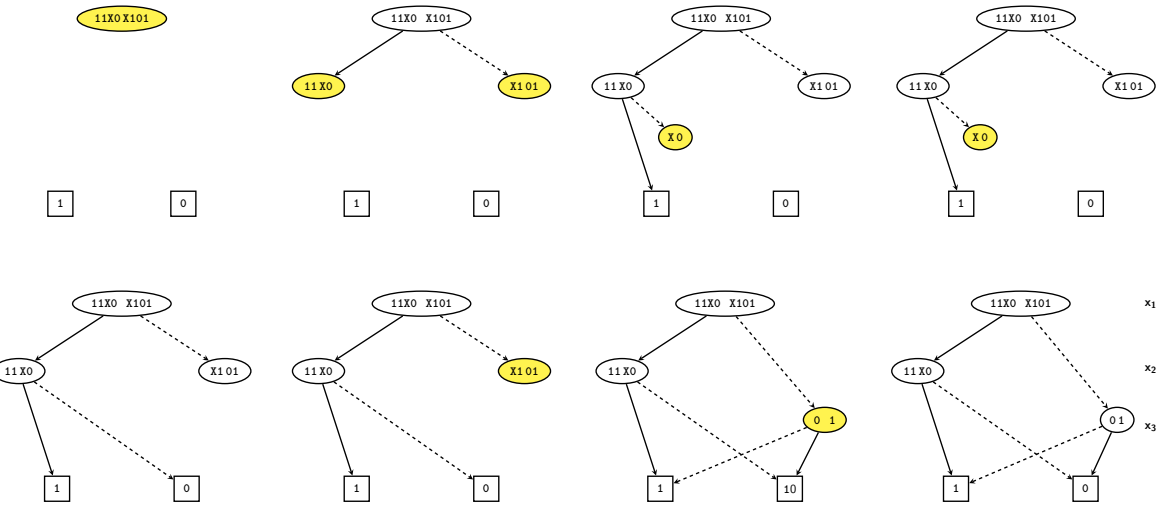


Don't Cares

- Eine Boolesche Funktion, die k „X“ enthält, steht eigentlich für 2^k unterschiedliche Funktionen, da jedes „X“ die Werte „0“ oder „1“ annehmen kann
- Für Funktionen in vielen Variablen und mit vielen „X“ sind das sehr viele unterschiedliche Funktionen, die man alle vereinfachen müsste, um die „kleinste“ Funktion zu finden
- Daher: Don't Cares in BDDs integrieren

Beispiel mit Beads

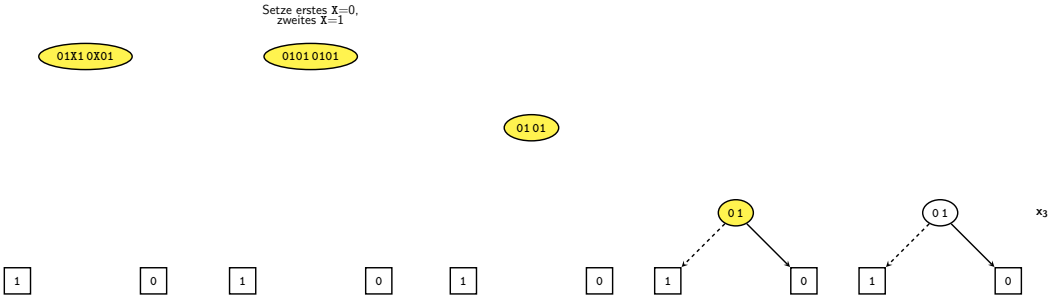
Halbierte Beads untereinander schreiben und durch Setzen der „X“ ein Square bilden;
Falls kein Square möglich: halbierte Beads in die nächste Ebene.



Noch ein Beispiel

Wir betrachten den Bead: `01X1 0X01`

1: `01X1` vs `0X01` 2: Setze `X`-Werte 3: Square 4:Bead 5: Beschriften



Fixieren aufschieben

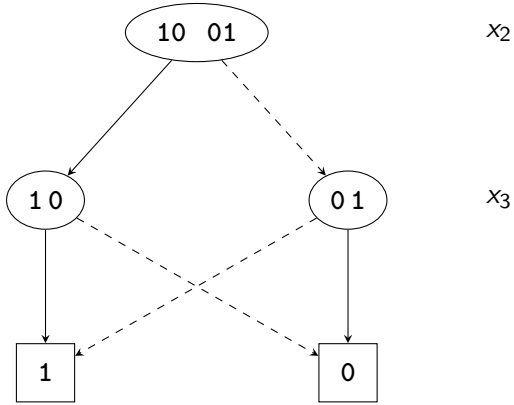
- Ein abschließendes Beispiel soll zeigen, dass es manchmal gut ist, die Werte von „Don't Care"-Stellen erst bei weiter unten liegenden Ebenen zu fixieren

Zwei Varianten:

- Gierige Variante: X so schnell wie möglich durch 0 oder 1 ersetzen
- Geduldige Variante: X so spät wie möglich durch 0 oder 1 ersetzen

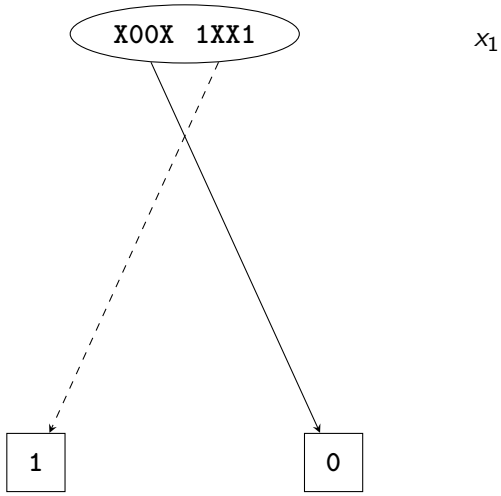
Beispiel (gierige Variante)

Bead: `X00X 1XX1`



Beispiel (geduldige Variante)

Bead: `X00X 1XX1`



Fixieren aufschieben

- Das voran gegangene Beispiel hat gezeigt, dass es manchmal möglich ist, eine bessere Vereinfachung zu erhalten, wenn man „Don't Cares“ „überleben“ lässt und erst später deren Werte fixiert
- Man benötigt also gute Heuristiken für „Don't Care“-Vereinfachungen, falls man BDDs mit „Don't Cares“ automatisiert minimieren will

Zusammenfassung Boolesche Algebra

- Boolesche Algebra
  - Algebra der Logik
  - Methode um boolesche Funktionen darzustellen
- Repräsentation Boolescher Funktionen
  - Formel
  - Wahrheitstabelle
  - If-Then-Else (ITE) Form
  - Binäre Entscheidungsdiagramme (BDDs)