1. Wahrheitswerte

Boole’sche Funktionen; der Typ `bool`; logische und relationale Operatoren; Kurzschlussauswertung.

Boole’sche Funktionen

- Boole’sche Funktion

$$f : \{0,1\}^2 \to \{0,1\}$$

- 0 entspricht „falsch”.
- 1 entspricht „wahr”.

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

Boole’sche Funktionen

- Boole’sche Funktion

$$f : \{0,1\}^2 \to \{0,1\}$$

- 0 entspricht „falsch”.
- 1 entspricht „wahr”.

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

Boole’sche Funktionen

- Boole’sche Funktion

$$f : \{0, 1\} \to \{0, 1\}$$

- 0 entspricht „falsch”.
- 1 entspricht „wahr”.

x	$\mathrm{NOT}(x)$
0	1
1	0

Vollständigkeit

- AND, OR und NOT sind die in $\mathrm{C{+}{+}}$ verfügbaren Boole'schen Funktionen.
- Alle anderen *binären* Boole'schen Funktionen sind daraus erzeugbar.

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

Vollständigkeit

- AND, OR und NOT sind die in $\mathrm{C{+}{+}}$ verfügbaren Boole'schen Funktionen.
- Alle anderen *binären* Boole'schen Funktionen sind daraus erzeugbar.

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

- $\mathrm{XOR}(x, y) = \mathrm{AND}(\mathrm{OR}(x, y), \mathrm{NOT}(\mathrm{AND}(x, y))$

Vollständigkeit Beweis

- Identifiziere binäre Boole'sche Funktionen mit ihrem charakteristischen Vektor.

Vollständigkeit Beweis

- Identifiziere binäre Boole'sche Funktionen mit ihrem charakteristischen Vektor.

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

Vollständigkeit Beweis

- Identifiziere binäre Boole'sche Funktionen mit ihrem charakteristischen Vektor.

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

Charakteristischer Vektor: 0110

Vollständigkeit Beweis

- Identifiziere binäre Boole'sche Funktionen mit ihrem charakteristischen Vektor.

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

Charakteristischer Vektor: 0110

$$\mathrm{XOR} = f_{0110}$$

Vollständigkeit Beweis

- Schritt 1: erzeuge die *elementaren* Funktionen $f_{0001}, f_{0010}, f_{0100}, f_{1000}$

$$
\begin{aligned}
f_{0001} &= \mathrm{AND}(x, y) \\
f_{0010} &= \mathrm{AND}(x, \mathrm{NOT}(y)) \\
f_{0100} &= \mathrm{AND}(y, \mathrm{NOT}(x)) \\
f_{1000} &= \mathrm{NOT}(\mathrm{OR}(x, y))
\end{aligned}
$$

Vollständigkeit Beweis

- Schritt 2: erzeuge alle Funktionen durch “Veroderung” elementarer Funktionen

$$f_{1101} = \mathrm{OR}(f_{1000}, \mathrm{OR}(f_{0100}, f_{0001}))$$

Vollständigkeit Beweis

- Schritt 2: erzeuge alle Funktionen durch "Veroderung" elementarer Funktionen

$$f_{1101} = \mathrm{OR}(f_{1000}, \mathrm{OR}(f_{0100}, f_{0001}))$$

- Schritt 3: erzeuge f_{0000}

$$f_{0000} = 0.$$

Der Typ `bool`

- Repräsentiert Wahrheitswerte.
- Literale **`true`** und **`false`**.
- Wertebereich $\{true, false\}$

```
bool b = true; // Variable mit Wert true
```

`bool` vs `int`: Konversion

- **`bool`** kann (leider) überall dort verwendet werden, wo **`int`** gefordert ist – und umgekehrt.

`bool` vs `int`: Konversion

- **`bool`** kann (leider) überall dort verwendet werden, wo **`int`** gefordert ist – und umgekehrt.

`bool`	$\rightarrow$	**`int`**
true	$\rightarrow$	1
false	$\rightarrow$	0

`bool` vs `int`: Konversion

- **`bool`** kann (leider) überall dort verwendet werden, wo **`int`** gefordert ist – und umgekehrt.

`bool`	$\rightarrow$	**`int`**
true	$\rightarrow$	1
false	$\rightarrow$	0

`int`	$\rightarrow$	**`bool`**
$\neq$**`0`**	$\rightarrow$	true
`0`	$\rightarrow$	false

`bool` vs `int`: Konversion

- **`bool`** kann (leider) überall dort verwendet werden, wo **`int`** gefordert ist – und umgekehrt.
- Viele existierende Programme verwenden statt **`bool`** oft nur die **`int`**-Werte 0 und 1. *Das ist schlechter Stil, der noch auf die Sprache C zurückgeht.*

`bool`	$\rightarrow$	**`int`**
true	$\rightarrow$	1
false	$\rightarrow$	0
`int`	$\rightarrow$	**`bool`**
$\neq$**`0`**	$\rightarrow$	true
`0`	$\rightarrow$	false

Logische Operatoren

	Symbol	Stelligkeit	Präzedenz	Assoziativität
Logisches Und (AND)	`&&`	2	6	links
Logisches Oder (OR)	`||`	2	5	links
Logisches Nicht (NOT)	`!`	1	16	rechts

`bool` ($\times$ **`bool`**) $\rightarrow$ **`bool`**

R-Wert ($\times$ R-Wert) $\rightarrow$ R-Wert

Relationale Operatoren

	Symbol	Stelligkeit	Präzedenz	Assoziativität
Kleiner	`<`	2	11	links
Grösser	`>`	2	11	links
Kleiner gleich	`<=`	2	11	links
Grösser gleich	`>=`	2	11	links
Gleich	`==`	2	10	links
Ungleich	`!=`	2	10	links

Zahlentyp $\times$ Zahlentyp $\rightarrow$ `bool`

R-Wert $\times$ R-Wert $\rightarrow$ R-Wert

DeMorgan’sche Regeln

- `!(a && b) == (!a || !b)`
- `!(a || b) == (!a && !b)`

DeMorgan'sche Regeln

- `!(a && b) == (!a || !b)`
- `!(a || b) == (!a && !b)`

! (reich *und* schön) == (arm *oder* hässlich)

Präzedenzen

Binäre Arithmetische Operatoren

binden stärker als

relationale Operatoren

und diese binden stärker als

binäre logische Operatoren.

```
7 + x < y && y != 3 * z
```

Präzedenzen

Binäre Arithmetische Operatoren

binden stärker als

relationale Operatoren

und diese binden stärker als

binäre logische Operatoren.

```
(7 + x) < y && y != (3 * z)
```

Präzedenzen

Binäre Arithmetische Operatoren

binden stärker als

relationale Operatoren

und diese binden stärker als

binäre logische Operatoren.

```
((7 + x) < y) && (y != (3 * z))
```

Kurzschlussauswertung

- Logische Operatoren `&&` und `||` werten den *linken Operanden zuerst* aus.
- Falls das Ergebnis dann schon feststeht, wird der rechte Operand *nicht mehr* ausgewertet.

```
x != 0 && z / x > y
```

Kurzschlussauswertung

- Logische Operatoren `&&` und `||` werten den *linken Operanden zuerst* aus.
- Falls das Ergebnis dann schon feststeht, wird der rechte Operand *nicht mehr* ausgewertet.

x hat Wert 1 $\Rightarrow$

```
x != 0 && z / x > y
```

Kurzschlussauswertung

- Logische Operatoren `&&` und `||` werten den *linken Operanden zuerst* aus.
- Falls das Ergebnis dann schon feststeht, wird der rechte Operand *nicht mehr* ausgewertet.

x hat Wert 1 $\Rightarrow$

```
true && z / x > y
```

Kurzschlussauswertung

- Logische Operatoren `&&` und `||` werten den *linken Operanden zuerst* aus.
- Falls das Ergebnis dann schon feststeht, wird der rechte Operand *nicht mehr* ausgewertet.

x hat Wert 1 $\Rightarrow$

```
true && z / x > y
```

Kurzschlussauswertung

- Logische Operatoren `&&` und `||` werten den *linken Operanden zuerst* aus.
- Falls das Ergebnis dann schon feststeht, wird der rechte Operand *nicht mehr* ausgewertet.

x hat Wert 0 $\Rightarrow$

```
x != 0 && z / x > y
```

Kurzschlussauswertung

- Logische Operatoren `&&` und `||` werten den *linken Operanden zuerst* aus.
- Falls das Ergebnis dann schon feststeht, wird der rechte Operand *nicht mehr* ausgewertet.

x hat Wert 0 $\Rightarrow$

```
false && z / x > y
```

Kurzschlussauswertung

- Logische Operatoren `&&` und `||` werten den *linken Operanden zuerst* aus.
- Falls das Ergebnis dann schon feststeht, wird der rechte Operand *nicht mehr* ausgewertet.

x hat Wert 0 $\Rightarrow$ `false`

Kurzschlussauswertung

- Logische Operatoren `&&` und `||` werten den *linken Operanden zuerst* aus.
- Falls das Ergebnis dann schon feststeht, wird der rechte Operand *nicht mehr* ausgewertet.

x hat Wert 0 $\Rightarrow$

```
x != 0 && z / x > y
```

$\Rightarrow$ Keine Division durch 0