2.6 Aussagenlogik: Schaltungen als Terme beschreiben

Leitidee

Eine Schaltung zu zeichnen dauert lange, und bei großen Schaltungen verliert man schnell den Überblick. Viel kürzer ist eine Formel wie $(\text{Tür} \wedge \text{Scharf}) \vee \text{Rauch}$. Solche Formeln gibt es schon viel länger als Computer: Die **Aussagenlogik** untersucht seit über 2000 Jahren, wie man Aussagen mit „und“, „oder“ und „nicht“ verknüpft. Der Ingenieur Claude Shannon hat erkannt, dass man mit genau diesen Formeln auch Schaltungen beschreiben kann. In diesem Abschnitt lernst du, wie solche Formeln – man nennt sie **Terme** – genau aufgebaut sind, wie man sie als **Baum** zeichnet und wie man zwischen Term und Schaltung hin- und herübersetzt.

Lernziele

Nach diesem Abschnitt kannst du

- die Begriffe Konjunktion, Disjunktion und Negation sachgemäß verwenden und ihre Belegungstabellen angeben,
- den Begriff des aussagenlogischen Terms induktiv definieren,
- entscheiden, ob eine Zeichenfolge ein Term ist, und die Klammerregeln anwenden,
- aussagenlogische Terme als Termbäume darstellen,
- logische Schaltungen durch Terme beschreiben und Terme in Schaltungen übertragen.

2.6.1 Aussagen und Aussagenvariablen

Eine **Aussage** ist ein Satz, der entweder wahr oder falsch ist. „Die Tür ist offen.“ ist eine Aussage. „Mach die Tür zu!“ ist keine Aussage, denn ein Befehl ist weder wahr noch falsch.

In der Aussagenlogik interessiert uns nicht, was eine Aussage genau sagt, sondern nur, ob sie wahr oder falsch ist. Deshalb ersetzt man sie durch einen Buchstaben oder ein kurzes Wort, zum Beispiel Tür für „Die Tür ist offen.“ Eine solche **Aussagenvariable** hat einen von zwei Werten: **1** für wahr, **0** für falsch.

Das passt genau zu den Schaltungen: Eine Leitung führt Strom (1) oder keinen Strom (0). Jeder Eingang einer Schaltung entspricht einer Aussagenvariablen. Wenn wir einer Variablen einen Wert geben, sagt man: Die Variable wird mit einem Wert **belegt**.

Definition: Aussagenvariable

Eine **Aussagenvariable** ist ein Platzhalter für eine Aussage. Sie kann mit einem der beiden Wahrheitswerte 1 (wahr) oder 0 (falsch) belegt werden. Man schreibt sie meist als Großbuchstaben: A, B, C …

Merke

Die **Aussagenlogik** beschreibt dieselben Zusammenhänge wie logische Schaltungen – nur mit Symbolen statt mit Gattern und Leitungen.

2.6.2 Die drei Verknüpfungen: Negation, Konjunktion, Disjunktion

Aus einfachen Aussagen baut man zusammengesetzte Aussagen. Dafür gibt es in der Aussagenlogik drei Zeichen. Du kennst sie schon von den Gattern:

- Die **Negation** $\neg A$ (gesprochen „nicht A“) dreht den Wert um. Sie entspricht dem NICHT-Gatter.
- Die **Konjunktion** $A \wedge B$ (gesprochen „A und B“) ist nur dann 1, wenn **beide** Teile 1 sind. Sie entspricht dem UND-Gatter.
- Die **Disjunktion** $A \vee B$ (gesprochen „A oder B“) ist 1, wenn **mindestens einer** der Teile 1 ist. Sie entspricht dem ODER-Gatter.

A	B	$\neg A$	$A \wedge B$	$A \vee B$
0	0	1	0	0
0	1	1	0	1
1	0	0	0	1
1	1	0	1	1

Definition: Negation, Konjunktion, Disjunktion

- Die **Negation** $\neg A$ ist genau dann 1, wenn A den Wert 0 hat.
- Die **Konjunktion** $A \wedge B$ ist genau dann 1, wenn A und B beide den Wert 1 haben.
- Die **Disjunktion** $A \vee B$ ist genau dann 1, wenn mindestens eine der beiden Variablen den Wert 1 hat.

Tipp

Eine Merkhilfe für die Zeichen: $\vee$ sieht aus wie ein „v“ – das lateinische Wort *vel* heißt „oder“. Das andere Zeichen $\wedge$ ist dann das „und“. Für das Zeichen $\neg$ stell dir einen Haken vor, der den Wert umdreht.

Die folgende Tabelle fasst zusammen, wie die drei Verknüpfungen in der Aussagenlogik, bei den Gattern und in Python heißen:

Name	Zeichen	gesprochen	Gatter	Python
Negation	$\neg A$	nicht A	NICHT (NOT)	`not a`
Konjunktion	$A \wedge B$	A und B	UND (AND)	`a and b`
Disjunktion	$A \vee B$	A oder B	ODER (OR)	`a or b`

Merke

Entsprechung zu Schaltungen: $\wedge$ (Konjunktion) entspricht dem UND-Gatter, $\vee$ (Disjunktion) dem ODER-Gatter, $\neg$ (Negation) dem NICHT-Gatter.

Stolperstein

Das „oder“ der Aussagenlogik ist **kein** „entweder … oder“. $A \vee B$ ist auch dann 1, wenn **beide** Variablen 1 sind. Im Alltag meinen wir mit „oder“ manchmal „nur eins von beiden“ („Willst du Tee oder Kaffee?“). Das wäre XOR. Das Zeichen $\oplus$ für XOR gehört aber nicht zu den Zeichen der Aussagenlogik, die du gleich kennenlernst. Dass man XOR auch mit $\wedge$, $\vee$ und $\neg$ ausdrücken kann, hast du im Auftrag *Belegungstabelle zu einer Schaltung* gesehen. Ein Verfahren, das immer funktioniert, lernst du im nächsten Abschnitt.

2.6.3 Was ist ein Term? Die induktive Definition

Nicht jede Zeichenfolge ist eine sinnvolle Formel. Mit $A \wedge \vee B$ kann niemand etwas anfangen. Wir brauchen also eine genaue Regel, welche Zeichenfolgen erlaubt sind. Die erlaubten Formeln heißen **aussagenlogische Terme**.

Die Regel funktioniert wie ein Baukasten: Man fängt mit den kleinsten Bausteinen an, den Variablen. Dann gibt es Regeln, wie man aus fertigen Termen größere Terme baut. Jeder Term, den man so gebaut hat, darf wieder als Baustein verwendet werden. Eine solche Definition, die Schritt für Schritt vom Kleinen zum Großen aufbaut, nennt man **induktiv**.

Merke

Induktive Definition eines aussagenlogischen Terms:

1. Jede Aussagenvariable (z. B. A, B, C) ist ein Term.
2. Ist t ein Term, so ist auch $\neg t$ (**Negation**) ein Term.
3. Sind t_1 und t_2 Terme, so sind auch $(t_1 \wedge t_2)$ (**Konjunktion**, entspricht AND) und $(t_1 \vee t_2)$ (**Disjunktion**, entspricht OR) Terme.
4. Nichts anderes ist ein Term.

Die Buchstaben t, t_1 und t_2 stehen hier nicht für Variablen, sondern für **beliebige Terme** – kleine wie A oder große wie $(A \wedge \neg B)$. Regel 4 ist wichtig: Sie sagt, dass es keine anderen Terme gibt als die, die man mit den Regeln 1 bis 3 bauen kann.

Gymnasium Ottweiler

Beispiel

Wir bauen den Term $((A \wedge B) \vee \neg C)$ Schritt für Schritt auf:

Schritt	gebauter Term	Regel	gebaut aus
1	A, B, C	1	–
2	$(A \wedge B)$	3	$t_1 = A, t_2 = B$
3	$\neg C$	2	$t = C$
4	$((A \wedge B) \vee \neg C)$	3	$t_1 = (A \wedge B), t_2 = \neg C$

Weil jeder Schritt eine der Regeln benutzt, ist $((A \wedge B) \vee \neg C)$ ein Term.

Umgekehrt kann man bei einer Zeichenfolge prüfen, ob sie ein Term ist. Diese Zeichenfolgen sind **keine** Terme:

Zeichenfolge	Warum kein Term?
$A\wedge$	Für $\wedge$ braucht man zwei Terme, rechts fehlt einer.
$\vee AB$	Das Zeichen $\vee$ muss **zwischen** zwei Termen stehen.
$(A \vee B$	Die schließende Klammer fehlt.
$A\neg B$	$\neg$ steht vor einem Term, nicht zwischen zwei Termen.
$A + B$	Das Zeichen $+$ kommt in den Regeln nicht vor.

Klammern sparen

Nach der strengen Definition stehen um jede Konjunktion und jede Disjunktion Klammern. Bei großen Termen wird das unübersichtlich. Deshalb gibt es Vereinbarungen, wann man Klammern weglassen darf:

1. **Die äußersten Klammern** dürfen weg: $(A \wedge B) \vee \neg C$ statt $((A \wedge B) \vee \neg C)$.
2. **Das Zeichen $\neg$ bezieht sich nur auf das, was direkt dahinter steht.** $\neg A \wedge B$ bedeutet $(\neg A) \wedge B$. Soll $\neg$ für mehr gelten, braucht man Klammern: $\neg(A \wedge B)$.
3. **Bei lauter $\wedge$ (oder lauter $\vee$)** ist die Klammerung egal, das Ergebnis ist immer dasselbe. Man schreibt einfach $A \wedge B \wedge C$ statt $((A \wedge B) \wedge C)$.
4. **$\wedge$ bindet stärker als $\vee$** – so wie Punkt vor Strich. $A \wedge B \vee C$ bedeutet also $(A \wedge B) \vee C$.

Python hält sich übrigens an dieselbe Reihenfolge: zuerst `not`, dann `and`, dann `or`.

Tipp

Die letzte Vereinbarung ist leicht zu vergessen. Setze deshalb Klammern, sobald $\wedge$ und $\vee$ zusammen vorkommen. Schreibe also lieber $(A \wedge B) \vee C$ als $A \wedge B \vee C$. Zu viele Klammern schaden nie.

Stolperstein

$\neg(A \vee B)$ und $\neg A \vee B$ sind zwei verschiedene Terme! Beim ersten wird das Ergebnis von $A \vee B$ umgedreht. Beim zweiten wird nur A umgedreht. Setze für $A = 0$, $B = 1$ ein: $\neg(0 \vee 1) = \neg 1 = 0$, aber $\neg 0 \vee 1 = 1 \vee 1 = 1$.

Arbeitsauftrag 65: Term oder kein Term?

Entscheide bei jeder Zeichenfolge, ob sie nach der strengen induktiven Definition ein Term ist. Begründe kurz. Gib bei den Termen an, in welcher Reihenfolge sie mit den Regeln aufgebaut werden.

a) $(A \vee \neg B)$
b) $\neg\neg A$
c) $(A \wedge B\vee)$
d) $(\neg(A \vee B) \wedge C)$
e) $\wedge(A, B)$

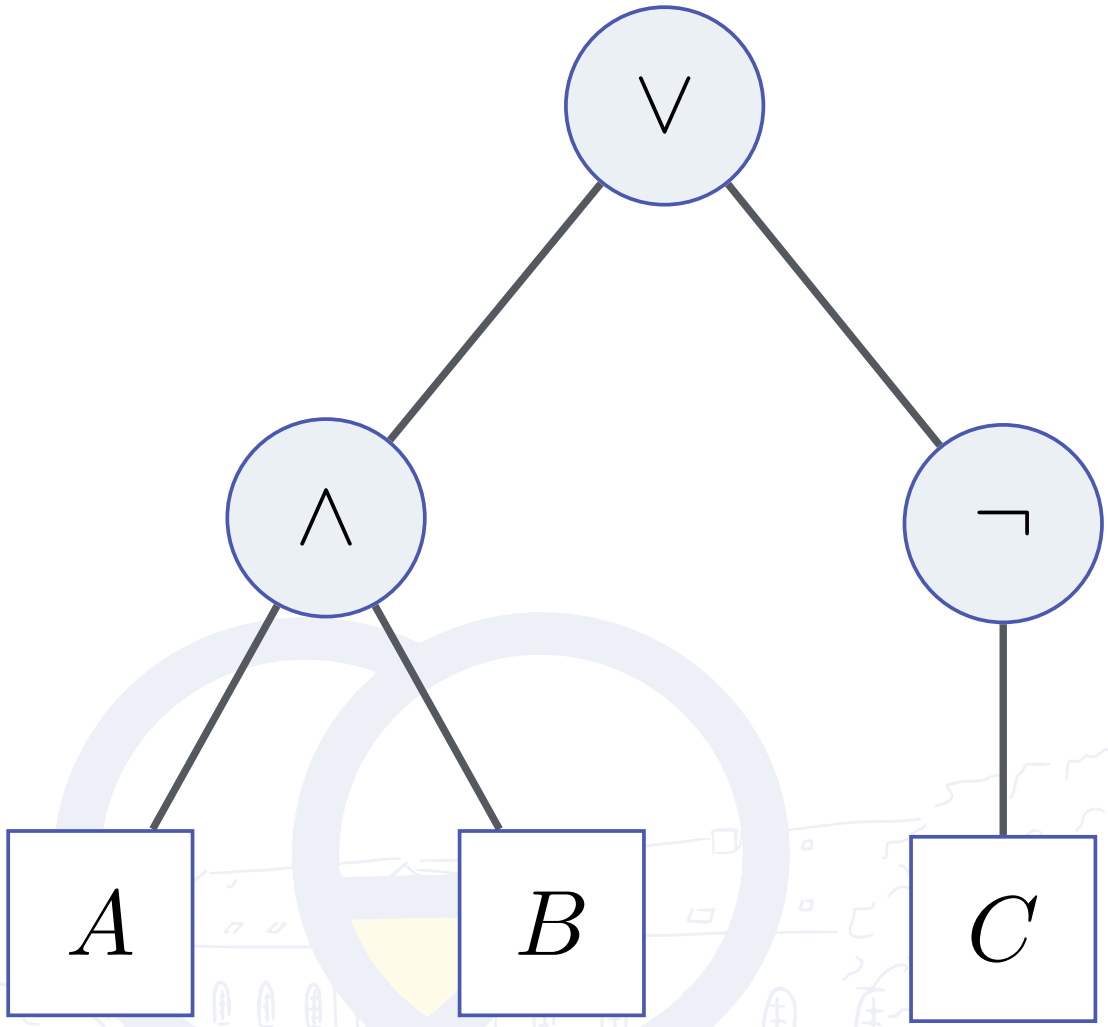
Meine Antworten:

2.6.4 Termbäume

Ein Term ist eine Zeichenkette. Seinen Aufbau sieht man aber besser in einem Bild. Dafür zeichnet man einen **Termbaum**. Er steht – anders als echte Bäume – auf dem Kopf:

- Ganz oben steht die **Wurzel**. Das ist der Operator, der **zuletzt** ausgerechnet wird.
- Unter jedem Operator hängen die Teilterme, mit denen er arbeitet: unter $\wedge$ und $\vee$ zwei, unter $\neg$ einer.
- Ganz unten stehen die **Blätter**. Das sind die Variablen.

Hier ist der Termbaum zum Term $(A \wedge B) \vee \neg C$:



Termbaum zu $(A \wedge B) \vee \neg C$: Der Operator, der zuletzt ausgewertet wird, steht ganz oben (Wurzel)

Vergleiche den Baum mit dem schrittweisen Aufbau im Beispiel oben: Jede Anwendung einer Regel ist ein Knoten im Baum. Die Variablen aus Schritt 1 sind die Blätter. Der letzte Schritt ($\vee$) ist die Wurzel.

Definition: Termbaum

Ein **Termbaum** stellt einen Term als Baumstruktur dar. Die **Blätter** sind die Aussagenvariablen, die **inneren Knoten** die Operatoren ($\neg$, $\wedge$, $\vee$). Der Operator, der zuletzt ausgewertet wird, steht an der Wurzel.

Merke

Ein **Termbaum** stellt einen Term als Baumstruktur dar: Die Blätter sind die Aussagenvariablen, die inneren Knoten die Operatoren ($\neg$, $\wedge$, $\vee$).

So zeichnest du einen Termbaum

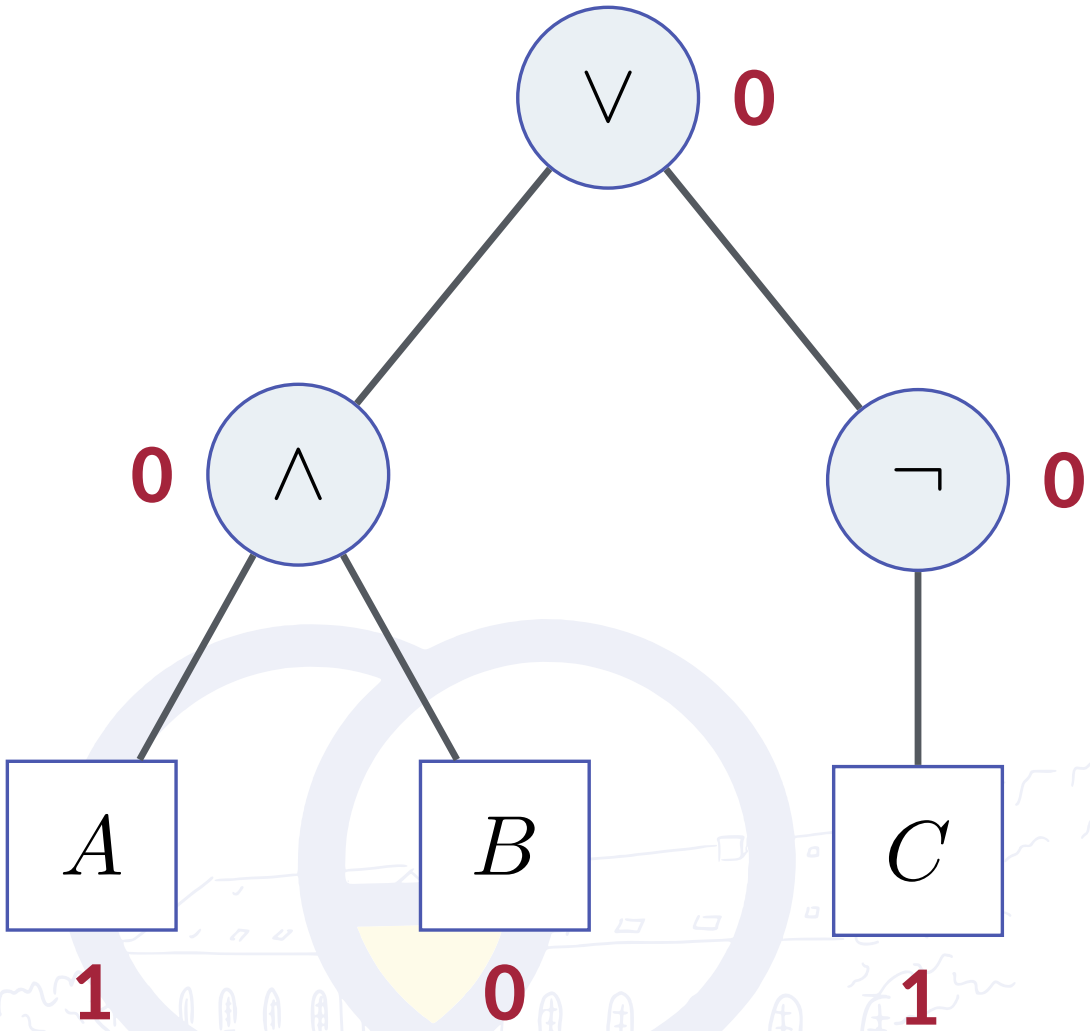
1. **Suche den Operator, der zuletzt ausgerechnet wird.** Das ist der Operator, der **außerhalb aller Klammern** steht. Stehen dort mehrere, nimm den schwächsten ($\vee$ vor $\wedge$ vor $\neg$). Er wird die Wurzel.
2. **Zeichne darunter seine Teilterme.** Bei $\wedge$ und $\vee$ ist das der Teil links und der Teil rechts vom Zeichen. Bei $\neg$ ist es das, was dahinter steht.
3. **Wiederhole** Schritt 1 und 2 für jeden Teilterm, bis nur noch Variablen übrig sind.

Für $(A \wedge B) \vee \neg C$: Außerhalb der Klammer stehen $\vee$ und $\neg$. Das schwächere Zeichen ist $\vee$, also ist $\vee$ die Wurzel. Links davon steht $(A \wedge B)$, rechts $\neg C$. Beide werden auf dieselbe Weise weiter zerlegt.

Einen Term am Baum ausrechnen

Am Termbaum kannst du auch gut ausrechnen, welchen Wert ein Term für eine bestimmte Belegung hat. Du rechnest **von unten nach oben**. Für $A = 1$, $B = 0$, $C = 1$ sieht das so aus (die Werte stehen rot neben den Knoten):

Gymnasium Ottweiler



Auswertung von $(A \wedge B) \vee \neg C$ für $A = 1, B = 0, C = 1$: Die Werte werden von den Blättern zur Wurzel weitergereicht

- $A \wedge B$: $1 \wedge 0 = 0$.
- $\neg C$: $\neg 1 = 0$.
- An der Wurzel: $0 \vee 0 = 0$. Der Term hat also für diese Belegung den Wert 0.

Arbeitsauftrag 66: Vom Term zum Termbaum

Zeichne den Termbaum zum Term

$$\neg(A \vee B) \wedge C.$$

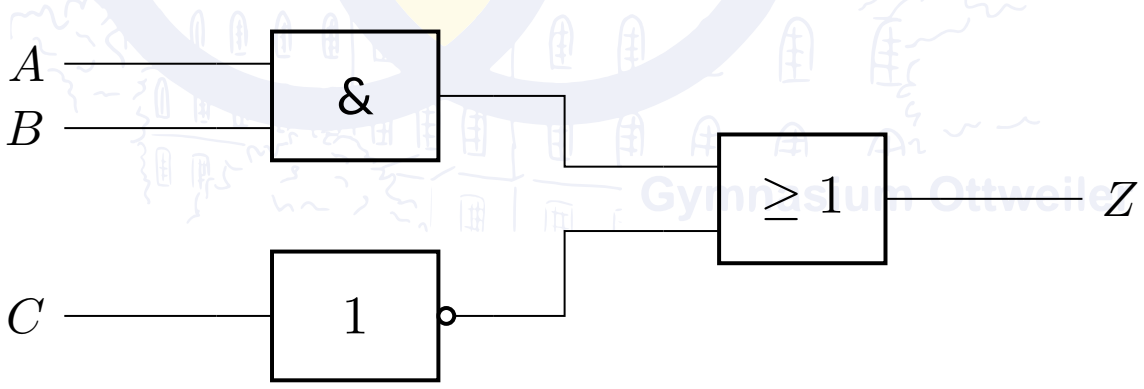
Mein Termbaum:

2.6.5 Vom Term zur Schaltung und zurück

Ein Termbaum und eine Schaltung haben denselben Aufbau. Dreh den Termbaum in Gedanken um 90 Grad, sodass die Wurzel rechts steht:

- Die **Blätter** (Variablen) werden zu den **Eingängen** links.
- Jeder **innere Knoten** wird zu einem **Gatter**: $\wedge$ zum UND-Gatter, $\vee$ zum ODER-Gatter, $\neg$ zum NICHT-Gatter.
- Die **Wurzel** wird zum letzten Gatter. Sein Ausgang ist der Ausgang der Schaltung.

So sieht die Schaltung zum Term $(A \wedge B) \vee \neg C$ aus:



Schaltung zum Term $(A \wedge B) \vee \neg C$: Jeder Knoten des Termbaums ist ein Gatter

Die Schaltung braucht so viele Gatter, wie der Termbaum innere Knoten hat – hier drei. Zum Bauen gehst du **von innen nach außen** vor: zuerst die Gatter für die innersten Teilterme ($A \wedge B$ und $\neg C$), zuletzt das Gatter der Wurzel.

Von der Schaltung zum Term

Der umgekehrte Weg geht genauso. Schreibe an jede Leitung, welchen Teilterm sie führt. Beginne an den Eingängen und arbeite dich nach rechts vor:

1. An den Eingängen stehen die Variablen.
2. Am Ausgang jedes Gatters steht der Teilterm, der aus seinen Eingängen entsteht. Ein UND-Gatter mit den Eingängen A und B liefert zum Beispiel $A \wedge B$.
3. Am Ausgang der Schaltung steht dann der gesamte Term.

Setze dabei um jeden Teilterm, der in ein weiteres Gatter läuft, Klammern. Überflüssige Klammern kannst du am Ende wieder weglassen.

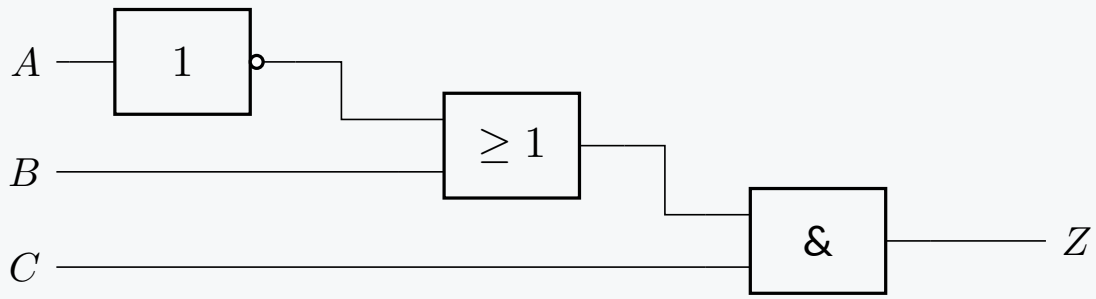
So kannst du es sagen

So sprichst du über Terme und Schaltungen – vom einfachen zum genauen Satz:

- *„Der Baum zeigt, wie der Term aufgebaut ist."* → *„Der Termbaum stellt den logischen Term dar und strukturiert ihn."*
- *„Die Formel ist die Schaltung."* → *„Der logische Term repräsentiert die Schaltung."*
- *„A ist 1 und B ist 0."* → *„Die logischen Variablen werden mit Werten belegt: A mit 1, B mit 0."*

Arbeitsauftrag 67: Schaltung beschreiben

Gib den Term an, den die folgende Schaltung darstellt. Zeichne anschließend den Termbaum.



Eine Schaltung mit drei Eingängen

Term und Termbaum:

Arbeitsauftrag 68: Termbaum der Alarmanlage

Im Abschnitt über Schaltungsentwurf hast du die Alarmanlage gebaut:

$$\text{Alarm} = (\text{Tür} \wedge \text{Scharf}) \vee \text{Rauch}$$

Zeichne den Termbaum zu diesem Term.

Mein Termbaum:

2.6.6 Terme mit Python kontrollieren

Terme lassen sich direkt als Python-Ausdrücke schreiben: `and` für $\wedge$, `or` für $\vee$, `not` für $\neg$. Damit kannst du prüfen, ob du einen Term richtig verstanden hast. Das folgende Programm gibt die Belegungstabelle zum Term $\neg(A \vee B) \wedge C$ aus:

```
1 def term1(a, b, c):
2     return int(not (a or b) and c)
3
4 print("Term 1: not (A or B) and C")
5 for a in [0, 1]:
6     for b in [0, 1]:
7         for c in [0, 1]:
8             print(f"A={a} B={b} C={c} -> {term1(a, b, c)}")
```

AUSGABE

```
Term 1: not (A or B) and C
A=0 B=0 C=0 -> 0
A=0 B=0 C=1 -> 1
A=0 B=1 C=0 -> 0
A=0 B=1 C=1 -> 0
A=1 B=0 C=0 -> 0
A=1 B=0 C=1 -> 0
A=1 B=1 C=0 -> 0
A=1 B=1 C=1 -> 0
```

- **Zeilen 1–2:** Die Funktion `term1` rechnet den Term für eine Belegung aus. Python wertet `not` vor `and` aus – genau wie in der Aussagenlogik. `int(...)` sorgt dafür, dass das Ergebnis als 0 oder 1 erscheint und nicht als `True` oder `False`.
- **Zeilen 5–7:** Drei ineinander geschachtelte Schleifen gehen alle Belegungen durch. Jede Variable läuft über die Liste `[0, 1]`. Die innerste Schleife (für `c`) läuft am schnellsten. So entstehen die $2^3 = 8$ Belegungen in der üblichen Reihenfolge von 000 bis 111.
- **Zeile 8:** Für jede Belegung wird eine Zeile der Tabelle ausgegeben.

Der Term ist nur bei $A = 0, B = 0, C = 1$ wahr. Das passt zur Bedeutung: „Weder A noch B, aber C.“

Arbeitsauftrag 69: Vom Term zur Schaltung

Jetzt geht es in die umgekehrte Richtung. Nimm den Term $\neg(A \vee B) \wedge C$ aus dem Auftrag *Vom Term zum Termbaum*.

1. Skizziere die zugehörige Schaltung mit IEC-Symbolen.
2. Baue die Schaltung in CircuitVerse: ein ODER-Gatter für $A \vee B$, ein NICHT-Gatter dahinter und ein UND-Gatter, das das Ergebnis mit C verknüpft.
3. Teste alle 8 Belegungen. Vergleiche mit deinem Termbaum und mit der Ausgabe des Python-Programms oben.

Meine Skizze:

Arbeitsauftrag 70: Genau eine von drei

Formuliere einen aussagenlogischen Term für folgende Situation: „Der Ausgang ist 1, wenn **genau eine** der drei Variablen A, B, C den Wert 1 hat.“ Notiere den Term symbolisch **und** als Python-Funktion `term3(a, b, c)`. Prüfe deine Funktion mit drei geschachtelten Schleifen.

Tipp: Überlege, in welchen Fällen der Ausgang 1 sein soll. Beschreibe **jeden Fall** mit einer Konjunktion, die genau auf diesen Fall passt. Verbinde dann alle Fälle mit $\vee$.

Mein Term:

Gymnasium Ottweiler

2.6.7 Weitere Übungen

Hier kannst du das Gelernte weiter üben. Die Punkte im Titel zeigen, wie schwer eine Aufgabe ist: ●○○ leicht, ●●○ mittel, ●●● knifflig. Fang mit den leichten an.

Arbeitsauftrag 71: Drei Schreibweisen ●○○

Ergänze die Tabelle.

Name	Zeichen	Gatter	Python
Negation			
	$\wedge$		
		ODER (OR)	

Arbeitsauftrag 72: Termbäume zeichnen ●○○

Zeichne zu jedem Term den Termbaum.

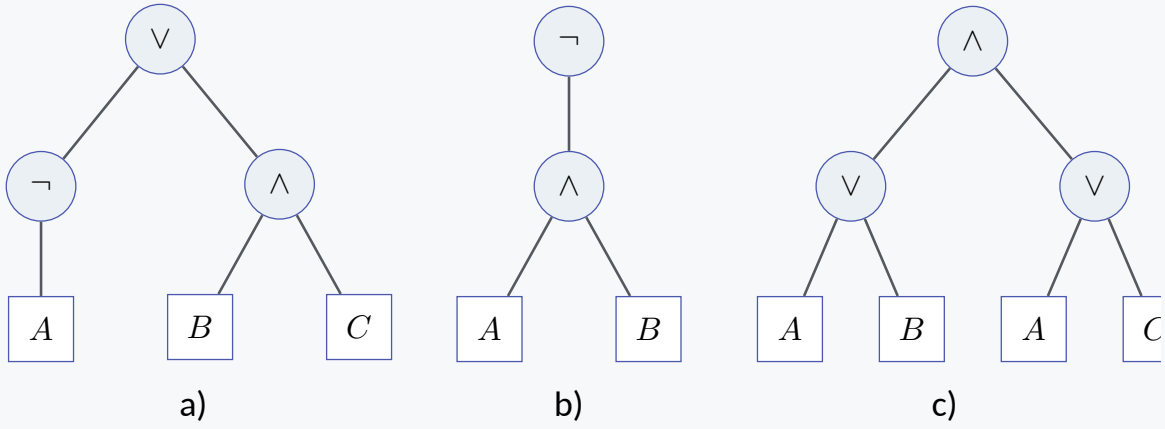
a) $\neg A \vee B$ b) $(A \vee B) \wedge C$ c) $\neg(A \vee \neg B)$

Deine Termbäume:

Gymnasium Ottweiler

Arbeitsauftrag 73: Vom Termbaum zum Term ●○○

Gib zu jedem Termbaum den Term an. Setze Klammern, wo sie nötig sind.



Drei Termbäume a), b) und c)

Deine Terme:

Arbeitsauftrag 74: Term oder kein Term? ●●○

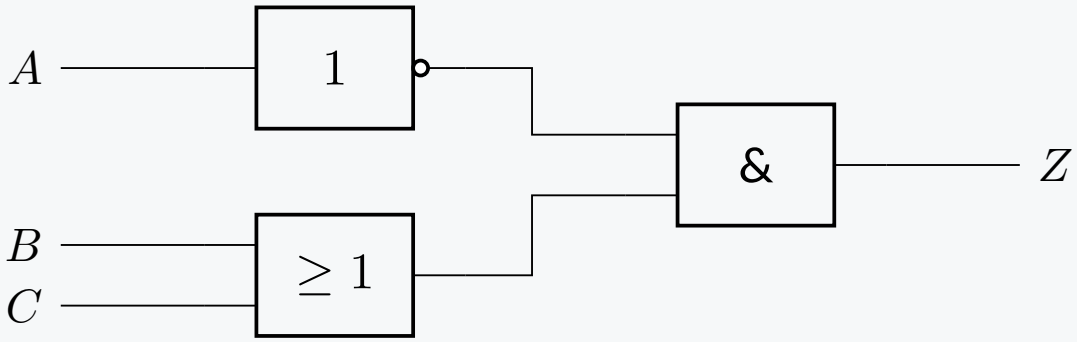
Entscheide mit der induktiven Definition, ob die Zeichenfolge ein aussagenlogischer Term ist. Begründe mit den Regeln.

a) A
b) $(A \wedge \neg C)$
c) $(A \vee B \vee)$
d) $((A \vee B) \wedge \neg C)$
e) $(A \oplus B)$
f) $\neg \wedge A$

Deine Antworten:

Arbeitsauftrag 75: Schaltung, Term und Termbaum ●●○

1. Gib den Term an, den die Schaltung darstellt.
2. Zeichne den Termbaum.
3. Berechne den Wert von Z für die Belegung $A = 0$, $B = 1$, $C = 0$. Schreibe die Zwischenwerte an die Knoten deines Termbaums.



Eine Schaltung mit drei Gattern

Term, Termbaum und Wert:

Arbeitsauftrag 76: Genau zwei von drei ●●●

Formuliere einen aussagenlogischen Term für die Situation: „Der Ausgang ist 1, wenn **genau zwei** der drei Variablen A, B, C den Wert 1 haben." Notiere den Term symbolisch **und** als Python-Funktion `genau_zwei(a, b, c)`. Prüfe die Funktion mit drei geschachtelten Schleifen.

Tipp

Geh vor wie im Auftrag *Genau eine von drei*: Überlege, welche Fälle es gibt. Beschreibe jeden Fall mit einer Konjunktion, in der alle drei Variablen vorkommen. Verbinde die Fälle mit $\vee$.

Dein Term:

2.6.8 Das Wichtigste in Kürze

- Eine **Aussagenvariable** steht für eine Aussage und hat den Wert 1 (wahr) oder 0 (falsch). Sie entspricht einem Eingang einer Schaltung.
- **Negation** $\neg$, **Konjunktion** $\wedge$ und **Disjunktion** $\vee$ entsprechen dem NICHT-, UND- und ODER-Gatter.
- Aussagenlogische Terme sind **induktiv** definiert: Variablen sind Terme; aus Termen entstehen mit $\neg$, $\wedge$, $\vee$ und Klammern neue Terme; sonst gibt es keine Terme.
- Klammern darf man sparen: äußerste Klammern weglassen, $\neg$ bindet am stärksten, dann $\wedge$, dann $\vee$.
- Ein **Termbaum** zeigt den Aufbau eines Terms: Blätter sind Variablen, innere Knoten Operatoren. Termbaum und Schaltung haben denselben Aufbau: Jeder innere Knoten ist ein Gatter.

Selbstcheck

- ☐ Ich kann Konjunktion, Disjunktion und Negation erklären und ihre Belegungstabellen angeben.
- ☐ Ich kann die induktive Definition eines aussagenlogischen Terms wiedergeben.
- ☐ Ich kann begründen, ob eine Zeichenfolge ein Term ist.
- ☐ Ich kann zu einem Term den Termbaum und die Schaltung zeichnen und zu einer Schaltung den Term angeben.
- ☐ Ich kann einen Term mit Python für alle Belegungen ausrechnen lassen.

Gymnasium Ottweiler