DS 20 – Klassische Aussagenlogik: Syntax und Termbäume

Informatik 10 – MINT-Zweig | Informatiksysteme

Lernziel

Ich beschreibe logische Schaltungen durch aussagenlogische Terme, definiere den Begriff des aussagenlogischen Terms induktiv und stelle Terme als Termbäume dar.

🕘 Stundenverlaufsplan (Doppelstunde, 90 Minuten)

Orientierungsrahmen für 11 SuS: Die ersten 80 Minuten bilden den Kern. Die letzten 10 Minuten sind bewusst flexibel und werden nicht mit neuem Pflichtstoff belegt.

Phase	Zeit	Inhalt/Aktion	Sozialform
Einstieg und Aktivierung	10'	Eine bekannte Schaltung zunächst in Worten beschreiben.	Plenum; kurze Einzeldenkzeit
Fragend-entwickelnde Erarbeitung	25'	Konjunktion, Disjunktion, Negation und induktiven Termaufbau aus Schaltungsteilen entwickeln.	Unterrichtsgespräch; Beiträge der SuS werden sichtbar festgehalten
Gemeinsame Umsetzung und Sicherung	20'	Einen Termbaum gemeinsam von innen nach außen konstruieren.	Plenum; Live-Coding, Tafelbild oder gemeinsame Simulation
Kurzer Transfer	15'	Einen gegebenen Term als Termbaum oder Schaltung darstellen.	Einzelarbeit mit anschließendem Tandemabgleich (4 Tandems + 1 Trio)
Auswertung	10'	Zusammenhang zwischen Teilterm, Knoten und Teilschaltung erläutern lassen.	Plenum; Lösung erklären und Fachsprache sichern
Reserve / Vertiefung	10'	Term für „genau eine von drei“ oder Python-Kontrolle.	Flexibel; bei Bedarf für Fragen, Fehlersuche oder Abschluss nutzen

🗣️ Sprachhilfe

- „Für die Eingangswerte … liefert die Schaltung den Ausgangswert …“
- „Der Term/die Tabelle repräsentiert die Schaltung, weil …“
- „Die Rückkopplung bzw. Gruppierung bewirkt …“

📌 Merksätze

- Die **Aussagenlogik** beschreibt dieselben Zusammenhänge wie logische Schaltungen – nur symbolisch statt schaltungstechnisch.
- **Induktive Definition eines aussagenlogischen Terms:**
  1. Jede **Aussagenvariable** (z. B. `A`, `B`, `C`) ist ein Term.
  2. Ist `t` ein Term, so ist auch `¬t` (**Negation**) ein Term.
  3. Sind `t1` und `t2` Terme, so sind auch `(t1 ∧ t2)` (**Konjunktion**, entspricht AND) und `(t1 ∨ t2)` (**Disjunktion**, entspricht OR) Terme.
  4. Nichts anderes ist ein Term.

- Ein **Termbaum** stellt einen Term als Baumstruktur dar: Die Blätter sind die Aussagenvariablen, die inneren Knoten die Operatoren (`¬`, `∧`, `∨`).
- Entsprechung zu Schaltungen: `∧` ↔ AND-Gatter, `∨` ↔ OR-Gatter, `¬` ↔ NOT-Gatter.

💡 Beispiel: Term zu Termbaum

Term: `(A ∧ B) ∨ ¬C`

Termbaum:



Dieser Term entspricht der Schaltung: `A` und `B` gehen in ein AND-Gatter, `C` geht in ein NOT-Gatter, deren Ausgänge gehen in ein OR-Gatter.

📝 Übung 1 – Term zu Termbaum

Zeichnet (auf Papier oder als ASCII-Skizze in dieser Zelle) den Termbaum zu folgendem Term:

`¬(A ∨ B) ∧ C`

(Euer Termbaum hier)

📝 Übung 2 – Schaltung zu Term

Ihr habt in DS 15/16 folgende Schaltung gebaut: `Alarm = (Tuer ∧ Scharf) ∨ Rauch`. Zeichnet dazu den Termbaum.

(Euer Termbaum hier)

🐍 Kontrolle: Terme als Python-Funktionen

Terme lassen sich direkt als Python-Ausdrücke mit `and`, `or`, `not` schreiben. Nutzt das, um eure Termbäume auf Plausibilität zu prüfen.

```python
from itertools import product

def term1(a, b, c):
    # ¬(A ∨ B) ∧ C
    return (not (a or b)) and c

def term2(tuer, scharf, rauch):
    # (Tuer ∧ Scharf) ∨ Rauch
    return (tuer and scharf) or rauch

print("--- Term 1: ¬(A ∨ B) ∧ C ---")
for a, b, c in product([0, 1], repeat=3):
    print(f"A={a} B={b} C={c} -> {int(term1(a,b,c))}")
```

```
--- Term 1: ¬(A ∨ B) ∧ 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
```

📝 Übung 3 – Vom Term zur Schaltung

Baut nun die **umgekehrte Richtung**: Nehmt den Term `¬(A ∨ B) ∧ C` aus Übung 1 (bzw. `term1` aus der Python-Kontrolle oben) und **baut die zugehörige Schaltung in CircuitVerse** (ein OR-Gatter für `A ∨ B`, ein NOT-Gatter dahinter, ein weiteres AND-Gatter mit `C`). Testet alle 8 Belegungen und vergleicht mit eurem Termbaum und der Python-Ausgabe von oben.

📝 Übung 4 – Eigenen Term formulieren

Formuliert einen aussagenlogischen Term für folgende Situation: „Der Ausgang ist `1`, wenn **genau eine** der drei Variablen `A`, `B`, `C` den Wert `1` hat." Notiert den Term symbolisch **und** als Python-Funktion.

```python
# TODO: Übung 4 - Term und Python-Funktion für "genau eine von drei"
def term3(a, b, c):
    pass
```

Lösung

✅ Lösung Übung 4

Lösung

```python
def term3(a, b, c):
    # (A ∧ ¬B ∧ ¬C) ∨ (¬A ∧ B ∧ ¬C) ∨ (¬A ∧ ¬B ∧ C)
    return (a and not b and not c) or (not a and b and not c) or (not a and not b and c)

for a, b, c in product([0, 1], repeat=3):
    print(f"A={a} B={b} C={c} -> {int(term3(a,b,c))}")
```

```
A=0 B=0 C=0 -> 0
A=0 B=0 C=1 -> 1
A=0 B=1 C=0 -> 1
A=0 B=1 C=1 -> 0
A=1 B=0 C=0 -> 1
A=1 B=0 C=1 -> 0
A=1 B=1 C=0 -> 0
A=1 B=1 C=1 -> 0
```