DS 22 – Karnaugh-Veitch-Diagramme und Minimierung

Informatik 10 – MINT-Zweig | Informatiksysteme

Lernziel

Ich vereinfache die DNF (mit bis zu vier Variablen) mithilfe von Karnaugh-Veitch-Diagrammen.

🕘 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 lange DNF und die zugehörige aufwendige Schaltung als Optimierungsproblem betrachten.	Plenum; kurze Einzeldenkzeit
Fragend-entwickelnde Erarbeitung	25'	Nachbarschaft und Blockbildung im KV-Diagramm durch Mustervergleich entwickeln.	Unterrichtsgespräch; Beiträge der SuS werden sichtbar festgehalten
Gemeinsame Umsetzung und Sicherung	20'	Ein Drei-Variablen-Beispiel gemeinsam eintragen, gruppieren und vereinfachen.	Plenum; Live-Coding, Tafelbild oder gemeinsame Simulation
Kurzer Transfer	15'	Ein weiteres Drei-Variablen-Diagramm bearbeiten.	Einzelarbeit mit anschließendem Tandemabgleich (4 Tandems + 1 Trio)
Auswertung	10'	Gruppenbildung und resultierenden vereinfachten Term begründen lassen.	Plenum; Lösung erklären und Fachsprache sichern
Reserve / Vertiefung	10'	Vier Variablen oder Alarmanlagen-Beispiel als Vertiefung.	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

- Ein **Karnaugh-Veitch-Diagramm (KV-Diagramm)** ordnet alle Belegungen so in einem Raster an, dass sich **benachbarte Felder nur in genau einer Variablen** unterscheiden.
- Ziel: Benachbarte `1`-Felder werden zu **Blöcken** der Größe 1, 2, 4, 8 … zusammengefasst. Je größer der Block, desto **weniger Variablen** braucht der zugehörige Term.
- Innerhalb eines Blocks fällt jede Variable weg, die sowohl **negiert als auch nicht-negiert** vorkommt – übrig bleiben nur die Variablen, die im gesamten Block **konstant** sind.
- Das KV-Diagramm liefert so eine **minimierte DNF**, die zur ursprünglichen DNF **äquivalent**, aber kürzer ist – wichtig für **kostengünstigere Schaltungen** (weniger Gatter).

💡 Beispiel: Minimierung von Hand

Belegungstabelle mit den `1`-Zeilen `A=0,B=0` und `A=0,B=1`:

DNF: `(¬A ∧ ¬B) ∨ (¬A ∧ B)`

Im KV-Diagramm liegen beide Felder nebeneinander (beide mit `A=0`) → sie lassen sich zu einem Block zusammenfassen, in dem `B` wegfällt:

Minimierte Form: `¬A`

Das lässt sich leicht mit Python nachprüfen – beide Terme müssen für alle Belegungen denselben Wert liefern:

```python
from itertools import product

def unminimiert(a, b):
    return (not a and not b) or (not a and b)

def minimiert(a, b):
    return not a

for a, b in product([0, 1], repeat=2):
    print(a, b, unminimiert(a, b) == minimiert(a, b))
```

```
0 0 True
0 1 True
1 0 True
1 1 True
```

📝 Übung 1 – KV-Diagramm zeichnen und minimieren

Gegeben ist folgende Belegungstabelle für drei Variablen:

A	B	C	Ausgang
0	0	0	1
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	0
1	1	1	0

1. Zeichnet auf Papier ein KV-Diagramm und findet die größtmöglichen Blöcke.
2. Notiert die minimierte Form.
3. Überprüft eure minimierte Form mit Python gegen die unminimierte DNF.

```python
def unminimiert_a(a, b, c):
    return ((not a and not b and not c) or (not a and not b and c) or
            (a and not b and not c) or (a and not b and c))

# TODO: Übung 1 - eure minimierte Form hier eintragen
def minimiert_a(a, b, c):
    pass

for a, b, c in product([0, 1], repeat=3):
    print(a, b, c, "->", unminimiert_a(a, b, c) == minimiert_a(a, b, c))
```

```
0 0 0 -> False
0 0 1 -> False
0 1 0 -> False
0 1 1 -> False
1 0 0 -> False
1 0 1 -> False
1 1 0 -> False
1 1 1 -> False
```

📝 Übung 2 – Von der Alarmanlage zurück zur Minimierung

Erinnert euch an die Alarmanlage aus DS 16: `Alarm = (Tuer ∧ Scharf) ∨ Rauch`. Ist dieser Term bereits minimal, oder lässt er sich mit einem KV-Diagramm weiter vereinfachen? Begründet eure Antwort in Worten (kein Python nötig).

(Eure Begründung hier)

🚀 Zusatzaufgabe – Vier Variablen

Erstellt selbst eine Belegungstabelle mit **vier** Variablen (`A, B, C, D`) und mindestens vier `1`-Zeilen, die sich zu einem einzigen Block zusammenfassen lassen. Prüft eure Minimierung mit Python.

Lösung

✅ Lösung Übung 1

Lösung

```python
def minimiert_a(a, b, c):
    # Block 1: A=0,B=0 (beide C) -> ¬A ∧ ¬B
    # Block 2: B=0 (beide A, beide C) -> ¬B  --> fasst sogar Block 1 mit ein!
    return not b

for a, b, c in product([0, 1], repeat=3):
    print(a, b, c, "->", unminimiert_a(a, b, c) == minimiert_a(a, b, c))
```

```
0 0 0 -> True
0 0 1 -> True
0 1 0 -> True
0 1 1 -> True
1 0 0 -> True
1 0 1 -> True
1 1 0 -> True
1 1 1 -> True
```