DS 21 – Belegungstabellen, Äquivalenz und Disjunktive Normalform (DNF)

Informatik 10 – MINT-Zweig | Informatiksysteme

Lernziel

Ich erstelle Belegungstabellen zu gegebenen aussagenlogischen Termen, erläutere den Begriff der Äquivalenz und erstelle die DNF zu gegebenen Termen bzw. Belegungstabellen.

🕘 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'	Die Ausgabe eines Terms für ausgewählte Belegungen vorhersagen.	Plenum; kurze Einzeldenkzeit
Fragend-entwickelnde Erarbeitung	25'	Aus den wahren Tabellenzeilen gemeinsam die Grundidee der DNF entdecken.	Unterrichtsgespräch; Beiträge der SuS werden sichtbar festgehalten
Gemeinsame Umsetzung und Sicherung	20'	Eine vollständige DNF aufbauen und Äquivalenz über gleiche Belegungstabellen sichern.	Plenum; Live-Coding, Tafelbild oder gemeinsame Simulation
Kurzer Transfer	15'	Zu einer kleinen Tabelle selbst eine DNF formulieren.	Einzelarbeit mit anschließendem Tandemabgleich (4 Tandems + 1 Trio)
Auswertung	10'	Jeden DNF-Baustein einer Tabellenzeile zuordnen und Äquivalenz erläutern lassen.	Plenum; Lösung erklären und Fachsprache sichern
Reserve / Vertiefung	10'	De-Morgan-Vergleich oder Python-Kontrolle 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

- Zwei aussagenlogische Terme heißen **äquivalent**, wenn sie für **jede** mögliche Belegung denselben Wahrheitswert liefern – also **dieselbe Belegungstabelle** besitzen. (Äquivalenz ist ein Begriff **über** Terme, kein eigener Operator!)
- Die **Disjunktive Normalform (DNF)** ist eine standardisierte Darstellung eines Terms:
  - Für **jede** Zeile der Belegungstabelle, in der der Ausgang `1` ist, bildet man ein **Minterm**: die Konjunktion aller Variablen, wobei Variablen mit Wert `0` negiert werden.
  - Alle Minterme werden anschließend **disjunktiv** (mit `∨`) verknüpft.
- Beispiel: Ist der Ausgang bei `A=1,B=0` und bei `A=0,B=1` gleich `1` (sonst `0`), lautet die DNF: `(A ∧ ¬B) ∨ (¬A ∧ B)` — das ist übrigens genau XOR!

💡 Beispiel: DNF aus Belegungstabelle ablesen

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

Zeilen mit Ausgang `1`: `(A=0,B=1)` und `(A=1,B=0)` → DNF: `(¬A ∧ B) ∨ (A ∧ ¬B)`

```python
from itertools import product

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

for a, b in product([0, 1], repeat=2):
    print(f"A={a} B={b} -> {int(dnf_beispiel(a, b))}")
```

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

📝 Vorübung – Von einem Term zur Belegungstabelle

Bevor ihr DNFs bestimmt: Erstellt **von Hand** die vollständige Belegungstabelle (3 Variablen, 8 Zeilen) zu folgendem Term:

`(A ∨ B) ∧ ¬C`

Überprüft euer Ergebnis anschließend mit der Python-Zelle unten.

```python
from itertools import product

def vorbuebung_term(a, b, c):
    return (a or b) and (not c)

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

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

📝 Übung 1 – DNF ablesen

Gegeben ist folgende Belegungstabelle:

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

Bestimmt die DNF (auf Papier) und überprüft sie anschließend mit Python.

```python
# TODO: Übung 1 - DNF als Python-Funktion eingeben und gegen die Tabelle prüfen
def meine_dnf(a, b, c):
    pass

for a, b, c in product([0, 1], repeat=3):
    print(f"A={a} B={b} C={c} -> {int(meine_dnf(a,b,c)) if meine_dnf(a,b,c) is not None else '?'}")
```

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

📝 Übung 2 – Äquivalenz prüfen

Prüft **automatisiert mit Python**, ob die beiden folgenden Terme äquivalent sind, indem ihr für **alle** Belegungen vergleicht, ob beide Funktionen stets denselben Wert liefern:

`Term A: ¬(A ∧ B)` und `Term B: ¬A ∨ ¬B` (De Morgansches Gesetz)

```python
def term_a(a, b):
    return not (a and b)

def term_b(a, b):
    return (not a) or (not b)

# TODO: Übung 2 - prüft für alle Belegungen, ob term_a und term_b immer gleich sind
aequivalent = True
for a, b in product([0, 1], repeat=2):
    # TODO: Vergleich einbauen und ggf. aequivalent = False setzen
    pass

print("Äquivalent:", aequivalent)
```

```
Äquivalent: True
```