ANSWERS

A8.3 Boolean identities and laws

Boolean algebra and logic circuits · A level · OCR H446 1.4.3, AQA 7517 4.6.5.1, Eduqas A500QS 1.2 · about 20 min

What this lesson is about

The identities, commutation, association, distribution, double negation and absorption, proved by truth table.

- The identities
- The laws
- Two ways to prove an equation
- Laws in robot code

Questions

6 marks in all

1. Which law is used in the step A ∧ (B ∨ C) = (A ∧ B) ∨ (A ∧ C)? [1 mark]

- ☐ **A** Distribution
- ☐ **B** Association
- ☐ **C** Commutation
- ☐ **D** Absorption

> **Answer:** A. Distribution multiplies out the bracket, like a × (b + c) = ab + ac.

2. What is A ∨ 1? [1 mark]

- ☐ **A** 1
- ☐ **B** A
- ☐ **C** 0
- ☐ **D** ¬A

> **Answer:** A. Anything OR true is true.

3. What does A ∨ (A ∧ B) simplify to? [1 mark]

- ☐ **A** A
- ☐ **B** A ∧ B
- ☐ **C** A ∨ B
- ☐ **D** B

> **Answer:** A. Absorption: if A is 1 the result is 1, and if A is 0 then A ∧ B is 0, so the result is A.

4. What does A ∨ (¬A ∧ B) simplify to? [1 mark]

- ☐ **A** A ∨ B
- ☐ **B** A
- ☐ **C** B
- ☐ **D** A ∧ B

> **Answer:** A. Distribute: (A ∨ ¬A) ∧ (A ∨ B) = 1 ∧ (A ∨ B) = A ∨ B.

5. Which of these are true for every value of A, B and C? [1 mark]

Tick every answer that is true.

- ☐ **A** $A \wedge \neg A = 0$
- ☐ **B** $A \vee B = B \vee A$
- ☐ **C** $A \vee A = 2A$
- ☐ **D** $A \wedge (A \vee B) = A$
- ☐ **E** $A \vee (B \wedge C) = (A \vee B) \wedge C$

> **Answer:** A, B, D. Complement, commutation and absorption hold; $A \vee A$ is A, and the last one fails at A = 1, C = 0.

6. What does this program print? [1 mark]

```
count = 0
for a in [0, 1]:
    for b in [0, 1]:
        for c in [0, 1]:
            if (a | (b & c)) != ((a | b) & c):
                count = count + 1
print(count)
```

> **Answer:**
>
> ```
> 2
> ```
>
> $A \vee (B \wedge C)$ and $(A \vee B) \wedge C$ differ whenever A = 1 and C = 0, which is 2 of the 8 rows.

The task: prove it by brute force

Write `equivalent(f, g)`. Its inputs `f` and `g` are functions of three bits, `f(a, b, c)`, each returning 0 or 1. It tries all eight rows in binary order (a, then b, then c, each 0 then 1) and returns the first row where `f` and `g` give different results, as a tuple `(a, b, c)`, or `None` if they agree on every row. Then, for each `(name, f, g)` in the list `laws`, in order, print one line: `<name>: equivalent` if they agree, or `<name>: not equivalent at A=<a> B=<b> C=<c>` using the row `equivalent` returned. That makes six lines.

```
# the two lines every program starts with: the commands, then the robot
from bugbot import *
connect()

laws = [
    ("distribution", lambda a, b, c: a & (b | c), lambda a, b, c: (a & b) | (a & c)),
    ("absorption", lambda a, b, c: a | (a & b), lambda a, b, c: a),
    ("association", lambda a, b, c: (a | b) | c, lambda a, b, c: a | (b | c)),
    ("OR over AND", lambda a, b, c: a | (b & c), lambda a, b, c: (a | b) & (a | c)),
    ("false friend", lambda a, b, c: a | (b & c), lambda a, b, c: (a | b) & c),
    ("half absorbed", lambda a, b, c: a & (a | b), lambda a, b, c: a | b),
]

def equivalent(f, g):
    return None
```

The hint students can ask for: Two expressions are equal only if they agree on every row, so loop through all eight combinations in binary order and stop at the first row where they differ. Return that row, or None if there is none, and let the printing code decide which message to show.

A solution

```
# the two lines every program starts with: the commands, then the robot
from bugbot import *
connect()

laws = [
    ("distribution", lambda a, b, c: a & (b | c), lambda a, b, c: (a & b) | (a & c)),
    ("absorption", lambda a, b, c: a | (a & b), lambda a, b, c: a),
    ("association", lambda a, b, c: (a | b) | c, lambda a, b, c: a | (b | c)),
    ("OR over AND", lambda a, b, c: a | (b & c), lambda a, b, c: (a | b) & (a | c)),
    ("false friend", lambda a, b, c: a | (b & c), lambda a, b, c: (a | b) & c),
    ("half absorbed", lambda a, b, c: a & (a | b), lambda a, b, c: a | b),
]

def equivalent(f, g):
    for a in [0, 1]:
        for b in [0, 1]:
            for c in [0, 1]:
                if f(a, b, c) != g(a, b, c):
                    return (a, b, c)
    return None

for name, f, g in laws:
    row = equivalent(f, g)
    if row is None:
        print(f"{name}: equivalent")
```

```
    else:
        print(f"{name}: not equivalent at A={row[0]} B={row[1]} C={row[2]}")
```

Any program that meets the task's checks is marked correct in the simulator; this is one way, not the only way.