Boolean logic

We are here

- Part 1: How computers works
  - Boolean logic, integers ⟵ TODAY
  - Instructions
  - Memory
- Part 2: Software development
  - Compiling (clang, make, …)
  - Architectures, portability (ABIs, …)
  - Code management (git)
- Part 3: Correctness
  - Specifications
  - Documentation, testing
  - Static & dynamic analysis, debugging
- Part 4: Performance
  - CPU pipelines, caches
  - Data structures
  - Parallel computation

Boolean values, operators, expressions

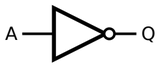
Boolean values

- `False = 0`
- `True  = 1`

Boolean variable:

$$x \in \{0, 1\}$$

Boolean operators

operator	math notation	pseudocode / Python	C code	logic gate
negation	¬	not	!, ~	
conjunction	∧, ×	and	&&, &	
disjunction	∨, +	or	||, |	

Boolean expressions

Example expression:

```
(a and b) or (not c)
```

Example function:

```
f(a, b, c) := (a and b) or (not c)
```

NOT operator

Truth table:

x	not x
0	1
1	0

Example assignment:

```
w := not a
```

AND operator

Truth table:

x	y	x and y
0	0	0
0	1	0
1	0	0
1	1	1

Example assignment:

```
z := a and (not b)
```

OR operator

Truth table:

x	y	x or y
0	0	0
0	1	1
1	0	1
1	1	1

Example assignment:

```
z := (not a) or (b and c)
```

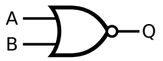
More operators!

XOR

x	y	x xor y
0	0	0
0	1	1
1	0	1
1	1	0

NAND

x	y	x nand y
0	0	1
0	1	1
1	0	1
1	1	0



NOR

x	y	x nor y
0	0	1
0	1	0
1	0	0
1	1	0

Q: How many distinct **unary** Boolean operators?

A: one?? (NOT)

Actually, we have 4 deterministic unary operators in total (counting 3 trivial unary operators):

always-false

x	0
0	0
1	0

always-true

x	1
0	1
1	1

identity

x	x
0	0
1	1

NOT

x	not x
0	1
1	0

Q: How many distinct binary operators?

A: As many as there are corresponding truth tables.

Q: How many distinct truth tables for two Boolean inputs and one Boolean output?

x	y	op(x, y)
0	0	?
0	1	?
1	0	?
1	1	?

A: $2^4 = 16$

Q: Why do we often use only NOT, AND, OR?

A: Because

- they are the most intuitive
- all nontrivial operators can be represented with NOT, AND, OR

Examples:

```
x nand y = not (x and y)
x xor y = (x or y) and (not (x and y))
```

Note: NAND and NOR are called *universal* logic gates:

every nontrivial operator can be represented with *only* NANDs (or *only* NORs)

Q: How do we prove this?

```
x xor y = (x or y) and (not (x and y))
```

A:

x	y	x xor y	(x or y) and (not (x and y))
0	0	0	0
0	1	1	1
1	0	1	1
1	1	0	0

The identity is correct iff the truth tables match.

Boolean identites and Boolean problems

Boolean identities I

- `x and 0 = 0`
- `x or 1 = 1`
- `x and 1 = x`
- `x or 0 = x`
- `x or x = x`
- `x and x = x`

Boolean identities II

- AND is commutative:

```
x and y = y and x
```

- AND is associative:

```
x and (y and z) = (x and y) and z
```

- OR is commutative:

```
x or y = y or x
```

- OR is associative:

```
x or (y or z) = (x or y) or z
```

Boolean identities III

- Distributivity (AND over OR):

```
x and (y or z) = (x and y) or (x and z)
```

- Distributivity (OR over AND):

```
x or (y and z) = (x or y) and (x or z)
```

- De Morgan's law (1):

```
(not x) and (not y) = not (x or y)
```

- De Morgan's law (2):

```
(not x) or (not y) = not (x and y)
```

Satisfiability problem

Given a Boolean expression, find a value for each variable such that the expression is true.

Equivalently: Find a 1 in the truth table.

Example: `x1 and ((not x2) or x3) and (not x3)`

x1	x2	x3	x1 and ((not x2) or x3) and (not x3)
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	0

Solution: `x1 = 1, x2 = 0, x3 = 0`

Definitions

- **Variable**: $x_j \in \{0, 1\}, \quad$ for some $j \in J \subseteq \mathbb{N}$

  Ex.:

```
x1
x5
```

- **Literal**: either $x_j \quad$ or $\quad \neg x_j, \quad$ for some $j \in J$

  Ex.:

```
x3
(not x8)
```

- **Disjunctive clause**: $\bigvee_{j \in J^0} \neg x_j \vee \bigvee_{j \in J^1} x_j \quad$ for some $J^0, J^1 \subseteq J$

  Ex.:

```
x2 or (not x4) or (not x6)
(not x1) or x5 or x6 or x7 or x9
```

- **Conjunctive clause**: $\bigwedge_{j \in J^0} \neg x_j \wedge \bigwedge_{j \in J^1} x_j \quad$ for some $J^0, J^1 \subseteq J$

  Ex.:

```
x2 and (not x4) and (not x6)
(not x1) and x5 and x6 and x7 and x9
```

Conjunctive normal form

The conjunctive normal form (CNF) is a conjunction of disjunctive clauses:

$$\bigwedge_{i \in I} \left(\bigvee_{j \in J^{i,0}} \neg x_j \vee \bigvee_{j \in J^{i,1}} x_j \right), \qquad \text{where } J^{i,0}, J^{i,1} \subseteq J \subseteq \mathbb{N},\ \forall i \in I \subseteq \mathbb{N}$$

Examples:

```
((x1 or x2) and (x3 or x4) and (x5 or x6))
```

```
((x1 or (not x2)) and (x3 or (not x4)))
```

```
    (x2 or (not x4) or (not x6))
and ((not x1) or x5 or x6 or x7 or x9)
and ((not x1) or (not x2) or (not x3))
and (x4 or x5 or x6)
```

Disjunctive normal form

The disjunctive normal form (DNF) is a disjunction of conjunctive clauses:

$$\bigvee_{i \in I} \left(\bigwedge_{j \in J^{i,0}} \neg x_j \wedge \bigwedge_{j \in J^{i,1}} x_j \right), \qquad \text{where } J^{i,0}, J^{i,1} \subseteq J \subseteq \mathbb{N},\ \forall i \in I \subseteq \mathbb{N}$$

Examples:

```
((x1 and x2) or (x3 and x4) or (x5 and x6))
```

```
((x1 and (not x2)) or (x3 and (not x4)))
```

```
   (x2 and (not x4) and (not x6))
or ((not x1) and x5 and x6 and x7 and x9)
or ((not x1) and (not x2) and (not x3))
or (x4 and x5 and x6)
```

Theorems

- Every Boolean expression can be put into CNF
  - For every Boolean expression with n variables and k literals using operators { NOT, AND, OR }, there exists an equivalent CNF with $n + k$ variables $3k$ clauses and $7k$ literals at most.
  - Satisfiability for a CNF (“SAT”) is hard.

- Every Boolean expression can be put in DNF
  - For every Boolean expression with n variables and k literals using operators { NOT, AND, OR }, there exists an equivalent DNF with n variables and $n \times 2^n$ literals at most
  - Satisfiability for a DNF is trivial.

Example:

Find a value for `x1, x2, x3, x4, x5, x6` such that the following expression (in DNF) is true.

```
   (x2 and (not x4) and (not x6))
or ((not x1) and x5 and x6 and x7 and x9)
or ((not x1) and (not x2) and (not x3))
or (x4 and x5 and x6)
```

1. Take any clause, e.g. `(x2 and (not x4) and (not x6))`.
2. Set variables to appropriate value so that all literals are true:

   `x2 = 1,  x4 = 0,  x6 = 0`.
3. All other variables can take any value.
4. Done