Informatics 1

Computation and Logic

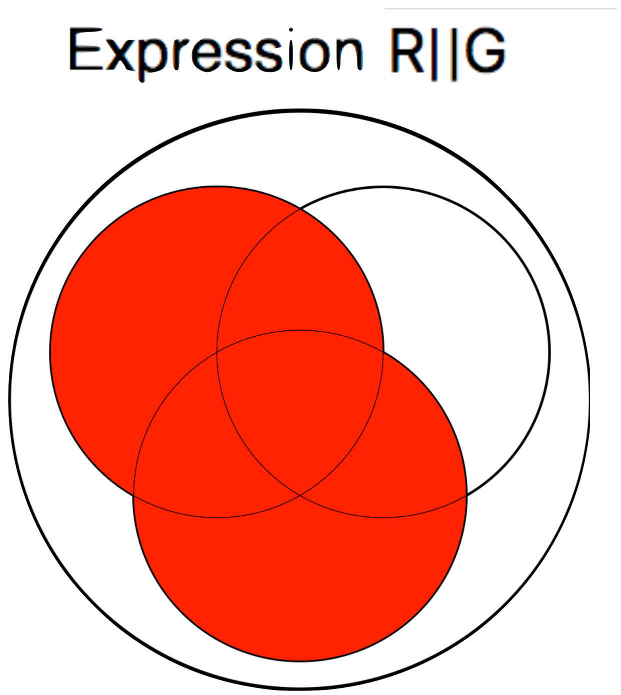
Boolean Algebra

Michael Fourman

www.inf.ed.ac.uk/teaching/courses/inf1/cl/tools/venn/

This is a tool for exploring Venn diagrams. The diagrams presented are randomly generated. You can change the first diagram (in the top left corner) by entering a Boolean expression using the three propositional letters `R A G`.

Your Boolean expression must be written in [Javascript notation](). As well as the three letters
and the Boolean operators, `||` (OR), `&&` (AND), and `!` (NOT),
you can use the constants `true` (TRUE) and `false` (FALSE).
You can also use the IF THEN ELSE, or ITE
[conditional operator]() *condition* `?` *expr1* `:` *expr2*
and also spaces, and parentheses `( )`.



Expression R||G

You can change the number of diagrams shown using a url such as

www.inf.ed.ac.uk/teaching/courses/inf1/cl/tools/venn/?n=16

This shows a square array with 16 x 16 = 256 diagrams.
16 is the largest value supported.

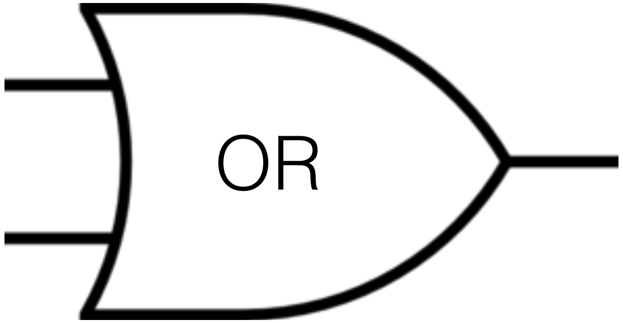
Basic Boolean operations

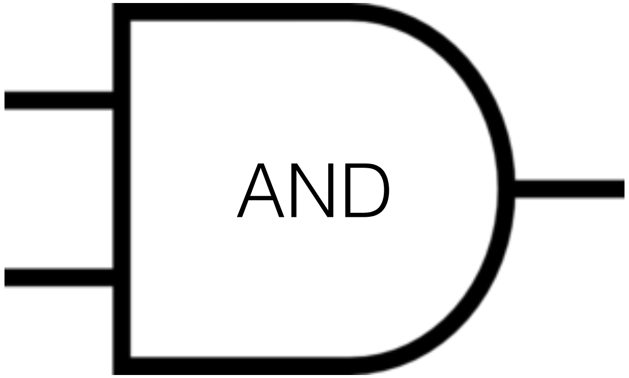
$\mathbf{1}, \top$	true, top
$\vee$	disjunction, or
$\wedge$	conjunction, and
$\neg$	negation, not
$\mathbf{0}, \bot$	false, bottom

Boole (1815 – 1864)

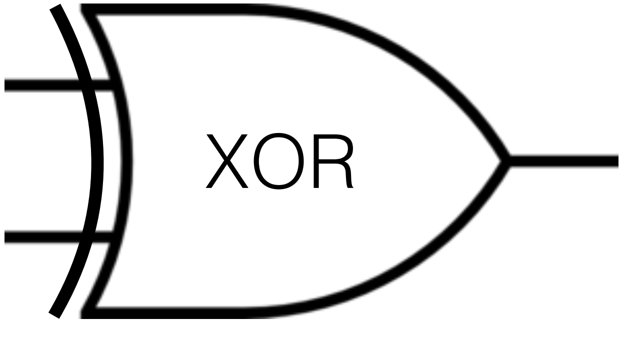
https://www.inf.ed.ac.uk/teaching/courses/inf1/cl/tools/venn/

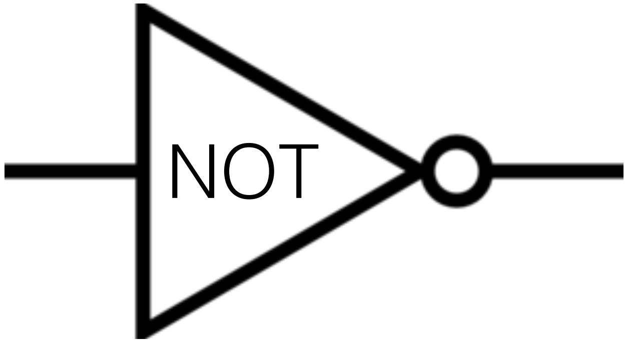
Syntax and circuits





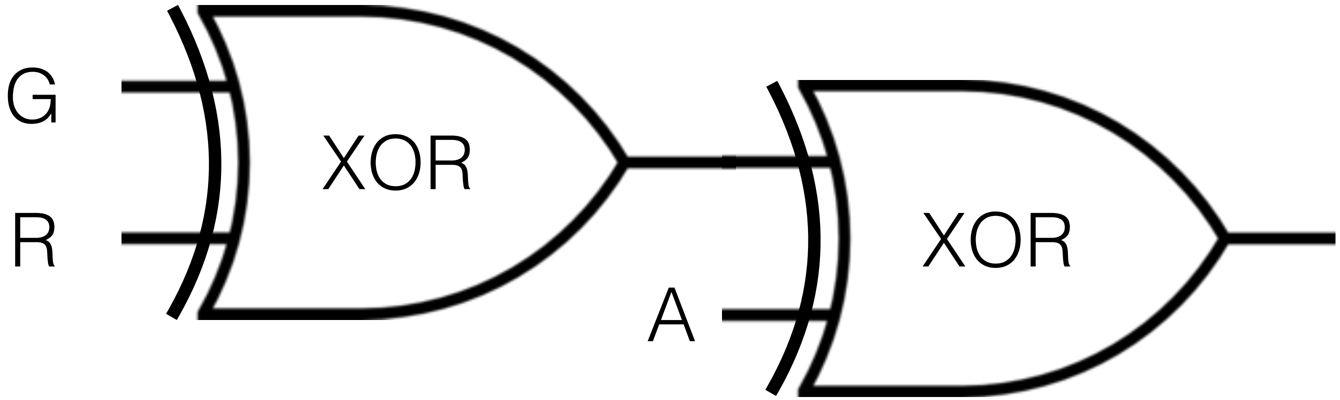
https://en.wikipedia.org/wiki/Combinational_logic

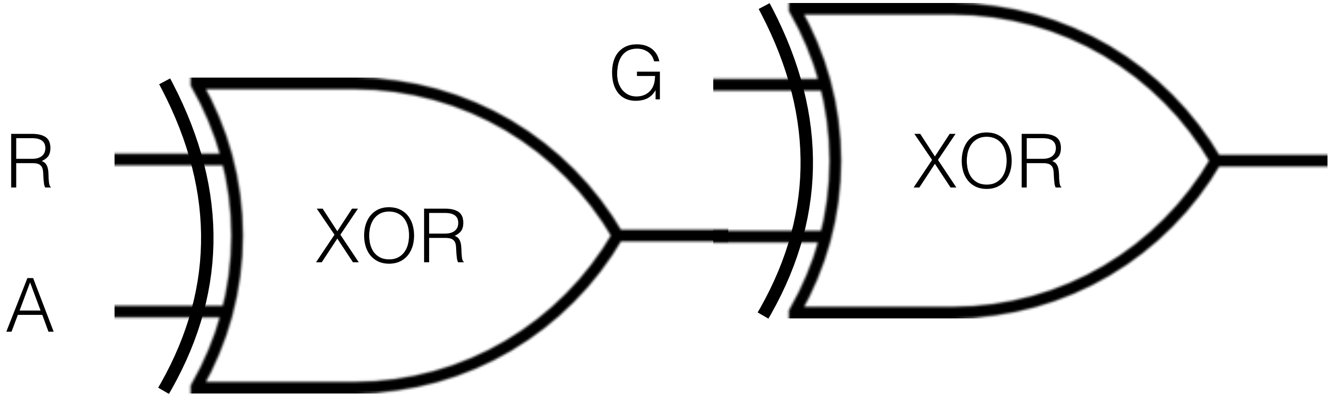




$$(G \oplus R) \oplus A \;=\; G \oplus (R \oplus A)$$

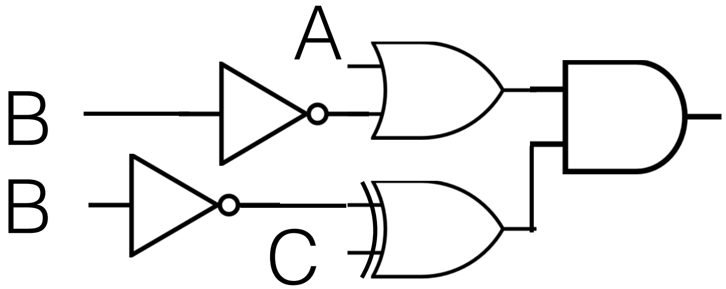
Syntax and circuits



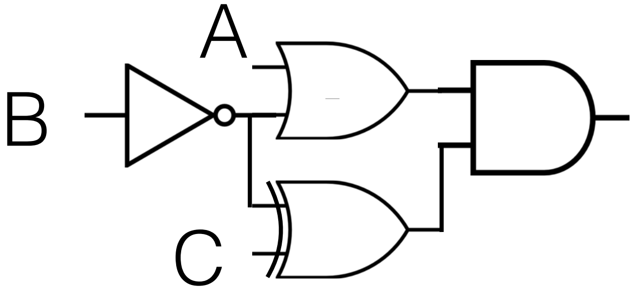


for any expression we can draw a circuit

`(A ∨ ¬B) && (¬B ⊕ C)`



but circuits allow us to share subexpressions



$(A \vee \neg B) \wedge (\neg B \oplus C)$

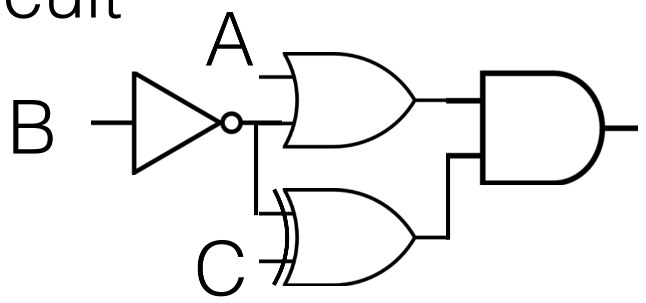
so does Haskell :
```
let bbar = not B
    in (A || bbar) && (bbar ⊕ C)
```

Exercise: define ⊕ in Haskell

Formula

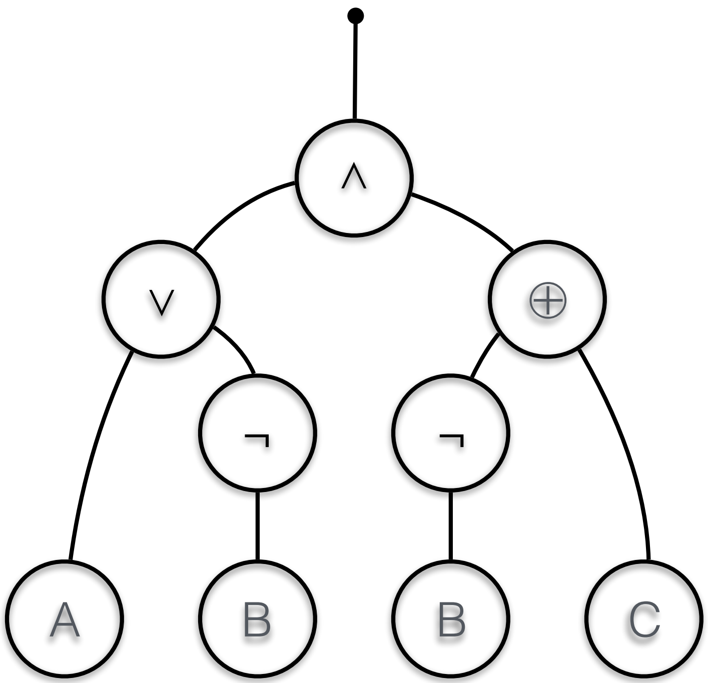
$(A \lor \neg B) \land (\neg B \oplus C)$

Circuit

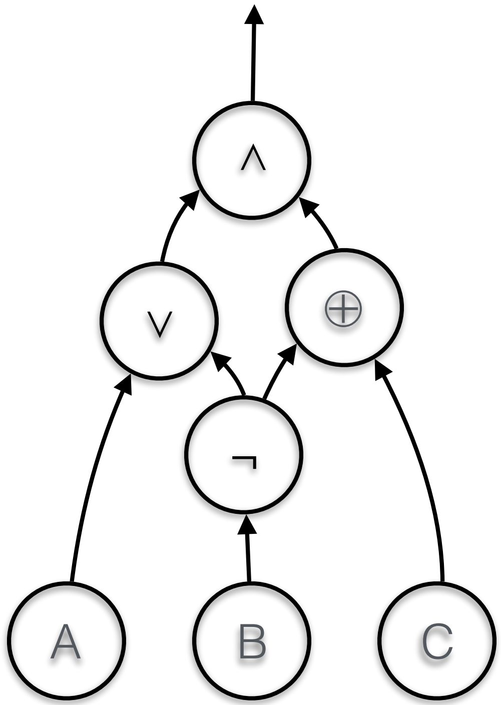


ABC	$A \lor \neg B$	$\neg B \oplus C$	out
000	1	1	1
001	1	0	0
010	0	0	0
011	0	1	0
100	1	1	1
101	1	0	0
110	1	0	0
111	1	1	1

Function (truth table)



Syntax tree



DAG

Computation (Haskell)

```
let bbar = not B
    in (A || bbar) && (bbar ⊕ C)
```

$\mathbb{Z}_2 = \{0, 1\}$ integers mod 2

+	0	1
0	0	1
1	1	0

×	0	1
0	0	0
1	0	1

–	
0	0
1	1

$$x \wedge y \equiv xy$$

$$x \vee y \equiv x + y - xy$$

$$\neg x \equiv 1 - x$$

Here, we use arithmetic mod 2
The same equations work if we use ordinary arithmetic!

https://en.wikipedia.org/wiki/Modular_arithmetic

∨	0	1
0	0	1
1	1	1

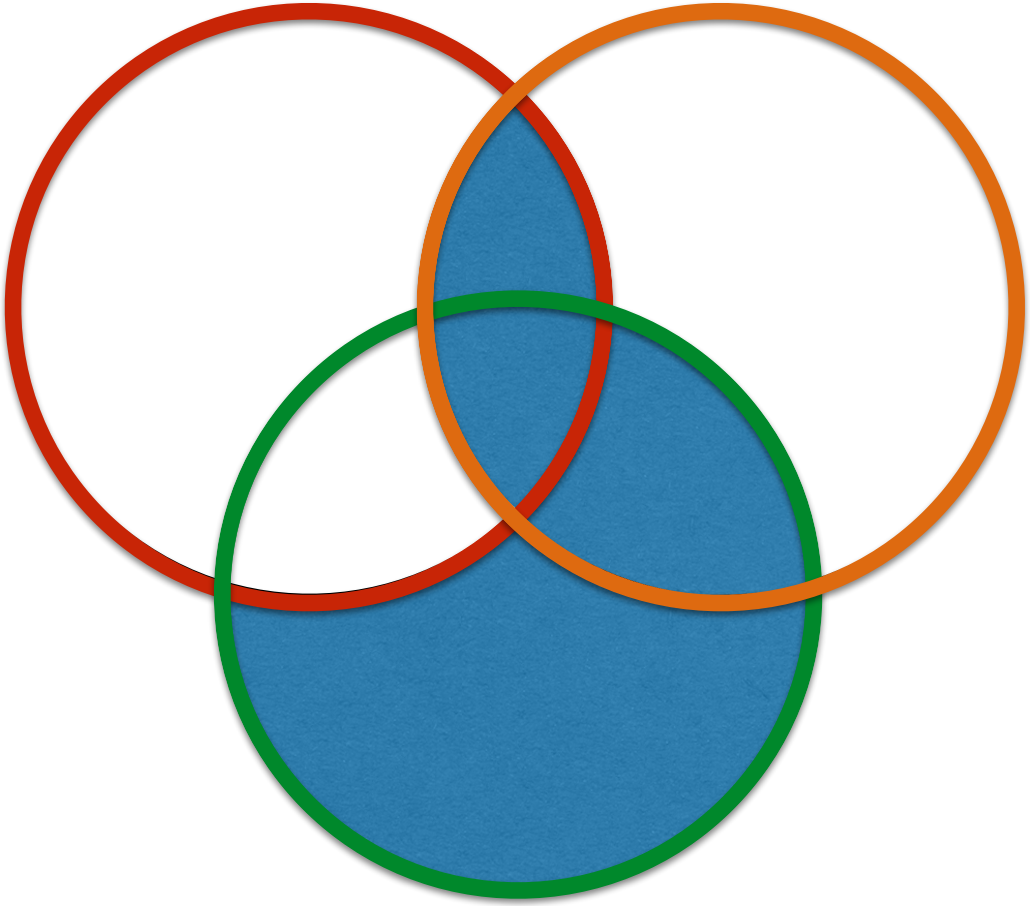
∧	0	1
0	0	0
1	0	1

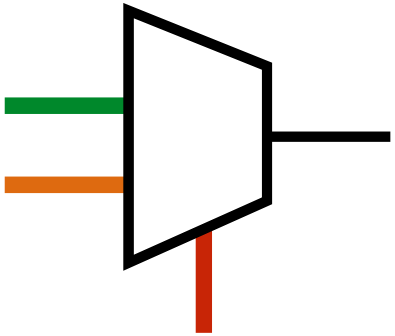
¬	
0	1
1	0

(R?A:G)

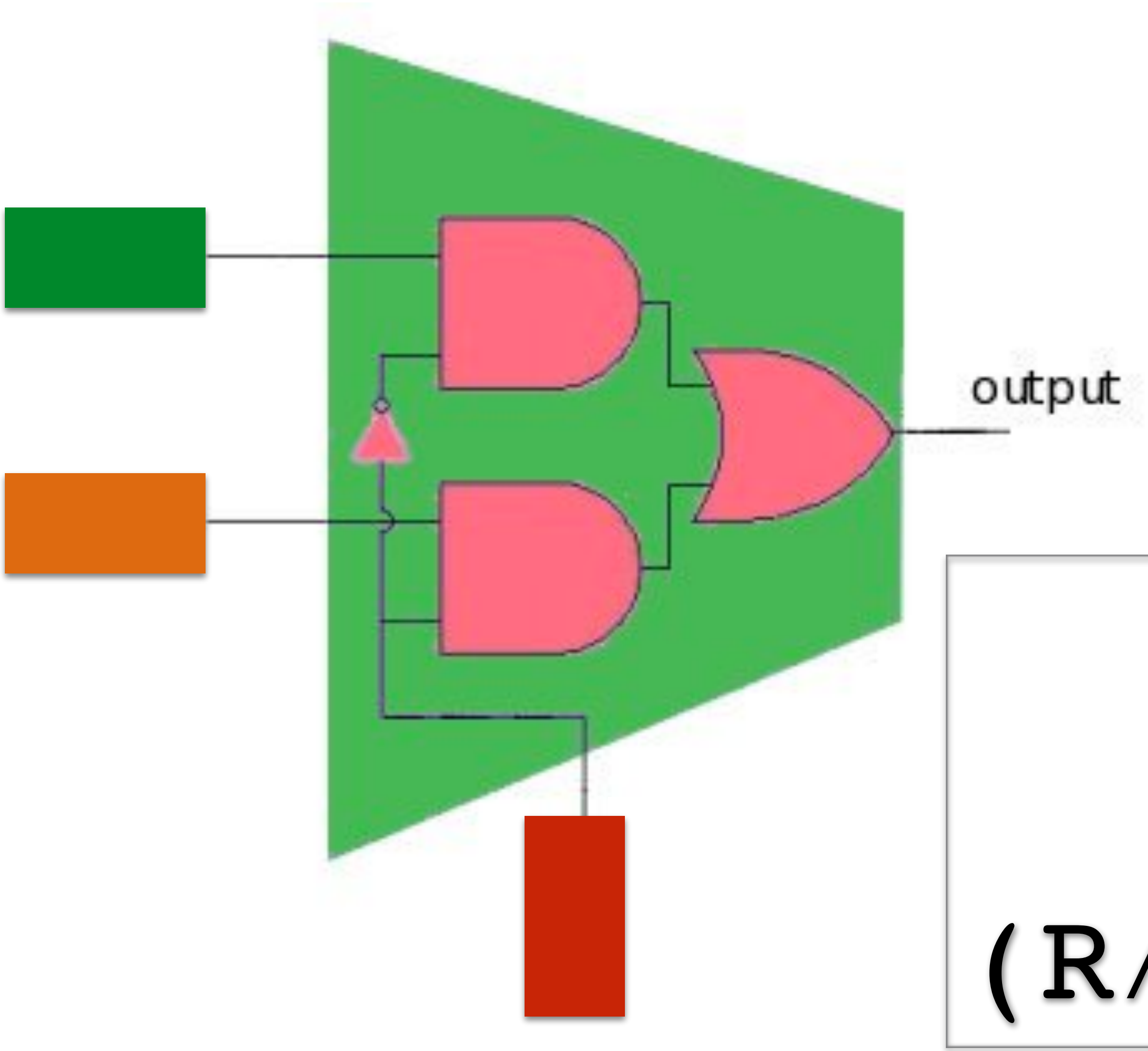
if R then A else G

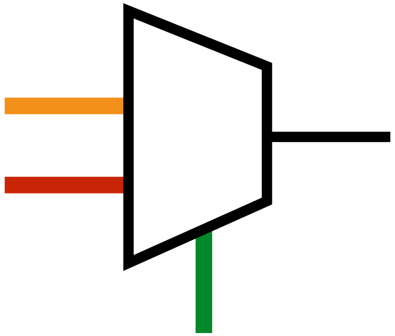
🟢	○	🔴	✗
🟢	🟠	🔴	✓
🟢	🟠	○	✓
○	○	○	✗
○	○	🔴	✗
○	🟠	🔴	✓
🟢	○	○	✓
○	🟠	○	✗





multiplexer – ITE





$$(R?A:G)$$

$$=$$

$$(R\land A)\lor(\neg R\land G)$$

if it is raining
then I will take an umbrella

$$R \rightarrow U$$

U

	→	0	1
R	0	?	?
	1	0	1

if R is true
and U is false
then
R → U is false

if it is raining
then I will take an umbrella

if $x < 2$
then $x < 4$

true for any value of x

$x < 4$

$x = 5$

$x = 2$

$x < 2$

→	0	1
0	?	?
1	0	1

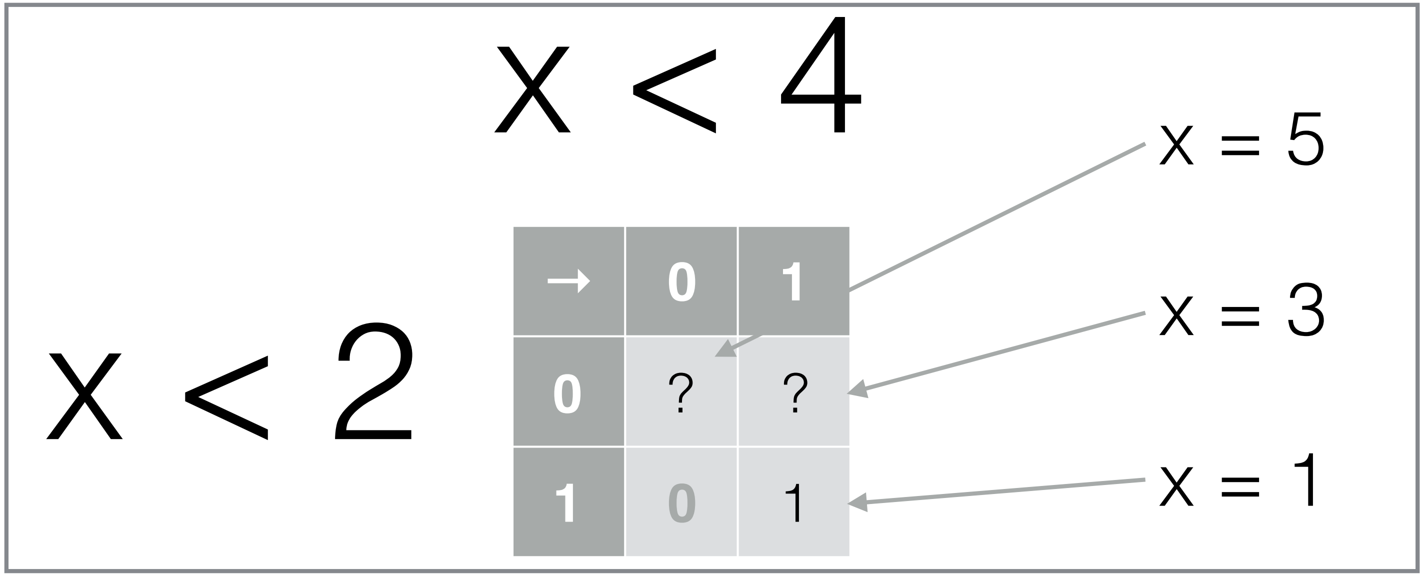
$x = 1$

if it is raining
then I will take an umbrella

if $x < 2$

then $x < 4$

true for any value of x
in particular
for $x = 1$, $x = 3$,and $x = 5$



$x < 4$

$x < 2$

→	0	1
0	?	?
1	0	1

$x = 5$

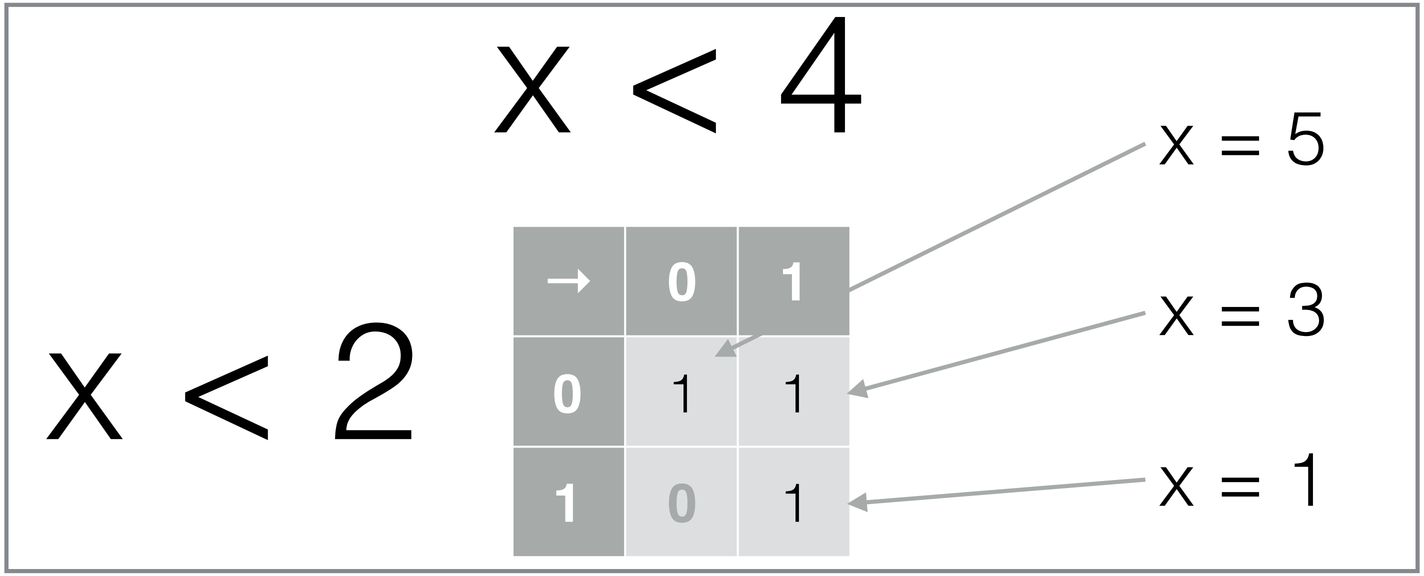
$x = 3$

$x = 1$

if it is raining
then I will take an umbrella

if $x < 2$
then $x < 4$

true for any value of x
in particular
for $x = 1$, $x = 3$,and $x = 5$



$x < 4$

$x < 2$

→	0	1
0	1	1
1	0	1

$x = 5$

$x = 3$

$x = 1$

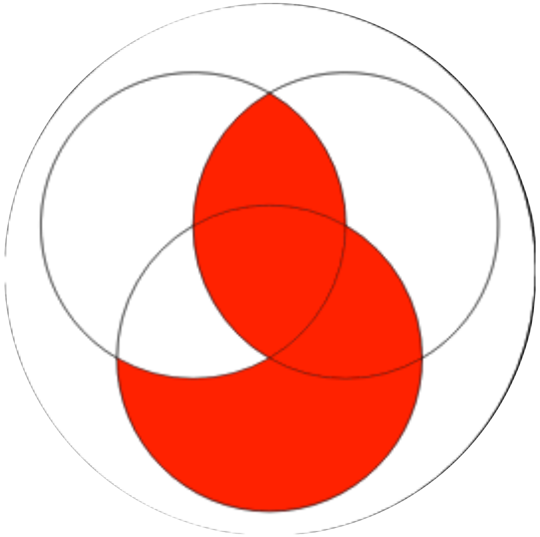
if it is raining
then I will take an umbrella

$$R \rightarrow U$$

U

	→	0	1
	0	1	1
R	1	0	1

R → U is true
unless we have
a counterexample
that makes R true
and U false



if A then B else C

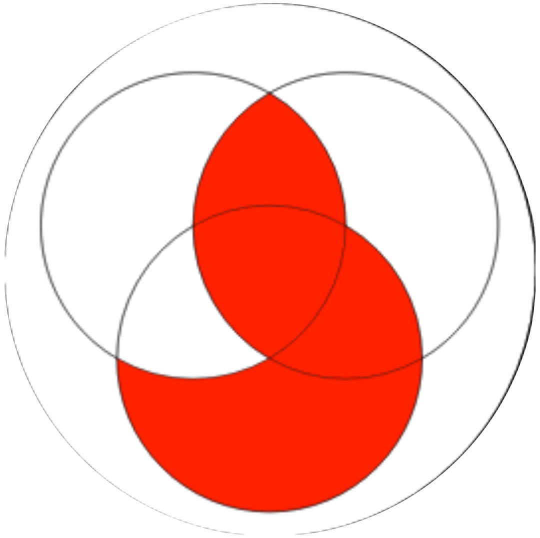
R?⊥:⊤

R?⊤:A

R?A:⊥

R?A:⊤

R?⊥:A



$R ? \bot : \top$	$\neg R$
$R ? \top : A$	$R \vee A$
$R ? A : \bot$	$R \wedge A$
$R ? A : \top$	$R \rightarrow A$
$R ? \bot : A$	$\neg(A \rightarrow R)$

Derived Operations

Definitions:

$$x \to y \equiv \neg x \lor y \qquad \text{implication}$$

$$x \leftarrow y \equiv x \lor \neg y$$

$$x \leftrightarrow y \equiv (\neg x \land \neg y) \lor (x \land y) \qquad \text{equality (iff)}$$

$$x \oplus y \equiv (\neg x \land y) \lor (x \land \neg y) \qquad \text{inequality (xor)}$$

Some equations:

$$x \leftrightarrow y = (x \to y) \land (x \leftarrow y)$$

$$x \oplus y = \neg(x \leftrightarrow y)$$

$$x \oplus y = \neg x \oplus \neg y$$

$$x \leftrightarrow y = \neg(x \oplus y)$$

$$x \leftrightarrow y = \neg x \leftrightarrow \neg y$$

Boolean Algebra

$\neg(a \to b) = a \wedge \neg b$ $\quad a \leftrightarrow b = (a \to b) \wedge (b \to a)$ $\quad a \to b = \neg a \vee b$

$\neg(a \vee b) = \neg a \wedge \neg b$ $\quad \neg(a \wedge b) = \neg a \vee \neg b$

$\neg 0 = 1$ $\quad \neg\neg a = a$ $\quad \neg 1 = 0$

$a \vee 1 = 1$ $\quad a \vee (b \wedge c) = (a \vee b) \wedge (a \vee c)$ $\quad a \wedge 0 = 0$

$a \vee 0 = a$ $\quad a \vee \neg a = 1$ $\quad a \wedge \neg a = 0$ $\quad a \wedge 1 = a$

an algebraic proof

$$\begin{aligned}
(x \leftrightarrow y) \leftrightarrow z &= \neg(x \leftrightarrow y) \leftrightarrow \neg z && (a \leftrightarrow b = \neg a \leftrightarrow \neg b) \\
&= (x \oplus y) \leftrightarrow \neg z && (\neg(a \leftrightarrow b) = a \oplus b) \\
&= (x \oplus y) \oplus z && (a \leftrightarrow \neg b = a \oplus b)
\end{aligned}$$