Some truth table examples

Importing some names with:

```
#import "../truthtable.typ": truth-table, l-and, l-or, l-imp, l-iff, l-not, l-var, l-operator, l-expr-tree
```

Import more as needed (do note, however, that there are several “private” functions, so importing everything at once might be excessive).

1. For $A \vee (B \to C)$:

```
#let expression = l-or("A", l-imp("B", "C"))

For #expression.repr:

#truth-table(expression)
```

A	B	C	$B \to C$	$A \vee (B \to C)$
T	T	T	T	T
F	T	T	T	T
T	F	T	T	T
F	F	T	T	T
T	T	F	F	T
F	T	F	F	F
T	F	F	T	T
F	F	F	T	T

It has the following tree:

```
#l-expr-tree(expression).flatten().map(c => c.repr).join([; ])
```

A; B; C; $B \to C$; $A \vee (B \to C)$

2. **Customize T/F:**

```
#let expression = l-iff("A", l-not(l-and("A", "A")))

#truth-table(expression, repr_true: 1, repr_false: 0)
```

A	$A \wedge A$	$\neg(A \wedge A)$	$A \leftrightarrow \neg(A \wedge A)$
1	1	0	0
0	0	1	0

3. **Skip a column:**

```
#let expression = l-iff("A", l-not(l-and("A", "A"), skip: true))

#truth-table(expression, repr_true: 1, repr_false: 0)
```

A	$A \wedge A$	$A \leftrightarrow \neg(A \wedge A)$
1	1	0
0	0	0

4. **Specify a custom text representation for your variables:** Use `l-var`:

```
#let expression = l-iff(l-var("P+Q", repr: $P(x) + Q(x)$), l-not(l-var("R", repr:
$R(x)$)))

#truth-table(expression, repr_true: 1, repr_false: 0)
```

$P(x) + Q(x)$	$R(x)$	$\neg R(x)$	$P(x) + Q(x) \leftrightarrow \neg R(x)$
1	1	0	0
0	1	0	1
1	0	1	1
0	0	1	0

5. **Customize the table:** Pass table parameters directly:

```
#let expression = l-iff("A", l-not(l-and("A", "A"), skip: true))

#truth-table(expression, repr_true: 1, repr_false: 0, fill: yellow, stroke: 5pt +
blue)
```

A	$A \wedge A$	$A \leftrightarrow \neg(A \wedge A)$
1	1	0
0	0	0

6. **Creating a custom operator:** Use `l-operator`. See the sample code below:

```
#let my-xor(a, b, skip: false) = {
    // convert bools and strings
    // to l-bool and l-var objects
    let a = l-logic-convert(a)
    let b = l-logic-convert(b)

    // automatically place parentheses around a and/or b
    // if they are composite expressions with 2+ children
    let a_repr = l-parens-repr-if-composite(a)
    let b_repr = l-parens-repr-if-composite(b)

    l-operator(
        "MY_XOR",
        a, b,
        value: mapping => {
            let a_val = (a.value)(mapping)  // consider the given map of (VARIABLE:
bool)
            let b_val = (b.value)(mapping)

            ((a_val or b_val) and not (a_val and b_val))
        },
        repr: $ #a_repr space dot(or) space #b_repr $,
        skip: skip
    )
}

#let expression = my-xor("A", "B")

#expression.repr

#align(center, truth-table(expression))
```

$A \dot{\vee} B$

A	B	$A \dot{\vee} B$
T	T	F
F	T	T
T	F	T
F	F	F