Propositional Logic

Demo

Vojtěch Bartík, March 2023

Logical Connectives

{NOT[a], AND[a, b], EQUIV[a, b], IMPLIES[a, b], NAND[a, b], NOR[a, b], OR[a, b], XOR[a, b]}

$\{\neg a,\ a \wedge b,\ a \Leftrightarrow b,\ a \Rightarrow b,\ a \uparrow b,\ a \downarrow b,\ a \vee b,\ a \oplus b\}$

$\{\neg a,\ a \wedge b,\ a \Leftrightarrow b,\ a \Rightarrow b,\ a \uparrow b,\ a \downarrow b,\ a \vee b,\ a \oplus b\}$

$\{\neg a,\ a \wedge b,\ a \Leftrightarrow b,\ a \Rightarrow b,\ a \uparrow b,\ a \downarrow b,\ a \vee b,\ a \oplus b\}$

FullForm /@ $\{\neg a,\ a \wedge b,\ a \Leftrightarrow b,\ a \Rightarrow b,\ a \uparrow b,\ a \downarrow b,\ a \vee b,\ a \oplus b\}$

{NOT[a], AND[a, b], EQUIV[a, b], IMPLIES[a, b], NAND[a, b], NOR[a, b], OR[a, b], XOR[a, b]}

{AND[a, b, c], OR[a, b, c], XOR[a, b, c]}

$\{a \wedge b \wedge c,\ a \vee b \vee c,\ a \oplus b \oplus c\}$

FullForm /@ $\{a \wedge b \wedge c,\ a \vee b \vee c,\ a \oplus b \oplus c\}$

{AND[a, b, c], OR[a, b, c], XOR[a, b, c]}

Attributes /@ Connectives

{{HoldAll, Protected}, {HoldAll, Protected}, {HoldAll, Protected}, {HoldAll, Protected}, {HoldAll, Protected}, {HoldAll, Protected}, {HoldAll, Protected}, {HoldAll, Protected}}

Logical Variables

Lower-case and capital letters of English alphabet: a, b, c, … , A, B, C, …
Symbols consisting of a single letter of English alphabet followed by a natural number: a1, b12, c123, … , A1, B12, C123, …
Lower-case letters of English alphabet subscripted with natural numbers: a_1, b_{12}, c_{123}, … , A_1, B_{12}, C_{123}, …

LAtomQ /@ {a, b, c, a1, b12, c123, A1, B12, C123, a_1, b_{12}, c_{123}, A_1, B_{12}, C_{123}}

{True, True, True, True, True, True, True, True, True, True, True, True, True, True, True}

LAtomQ /@ {aa, aa_1, a_b, α, A, $\boldsymbol{a}$, $\mathcal{A}$, $\mathfrak{a}$, $\mathfrak{A}$, $\mathbb{a}$, $\mathbb{A}$, False, FALSE, True, TRUE}

{False, False, False, False, False, False, False, False, False, False, False, False, False, False, False}

Logical Formulae

All logical variables are logical formulae.
If α is a logical formula then Hold[α], HoldForm[α] are also logical formulae.

```
LFormulaQ /@ {a, a1, a₁, False, FALSE, True, TRUE}
```

{True, True, True, True, True, True, True}

```
LFormulaQ /@ {Hold[a], Hold[a1], Hold[a₁], Hold[False], Hold[FALSE], Hold[True], Hold[TRUE]}
```

{True, True, True, True, True, True, True}

```
LFormulaQ /@ {HoldForm[a], HoldForm[a1], HoldForm[a₁],
  HoldForm[False], HoldForm[FALSE], HoldForm[True], HoldForm[TRUE]}
```

{True, True, True, True, True, True, True}

```
LFormulaQ[((x₁ ⇒ y↑x₂) ∧ z) ↓ ((x₂ ∨ x₃ ⇔ x) ⊕ (¬ x₁ ~NOR~ ¬ z))]
```

True

```
LFormulaQ[(HoldForm[x₁ ⇒ Hold[y↑x₂]] ∧ z) ↓ (HoldForm[x₂ ∨ x₃] ⇔ x ⊕ (¬ x₁ ⇒ ¬ z))]
```

True

```
Context /@ Connectives
```

{PropositionalLogic`, PropositionalLogic`, PropositionalLogic`, PropositionalLogic`, PropositionalLogic`, PropositionalLogic`, PropositionalLogic`, PropositionalLogic`}

Logical Strings (Formulae in Polish Notation)

Example 1

```
{s11 = "↓∧xx₁~NAND~x1¬y", LStringQ[s11]}
```

{↓∧xx₁~NAND~x1¬y, False}

```
LStringQ[s11, Trace → True]
```

False: Members of the list {xx₁} are not logical variables.

Example 2

```
{s21 = "↓∧xx₁~NAND~x1¬y", LStringQ[s21]}
```

{↓∧xx₁~NAND~x1¬y, True}

```
LStringQ[s21, Trace → True]
```

{↓ ∧ x x₁ ↑ x1 ¬ y}
{↓ ∧ x x$1 ↑ x1 ¬ y}
{NOR, AND, x, x₁, NAND, x1, NOT, y}
{NOR, AND, x, x₁, NAND, x1, True}
{NOR, AND, x, x₁, True}
{NOR, True, True}
{True}

Example 3

```
{s31 = "↓∧⇒x₁⇔yx1z12⊕⇒∨xx₃x↑¬x₁¬z", LStringQ[s31]}
```

{↓∧⇒x₁⇔yx1z12⊕⇒∨xx₃x↑¬x₁¬z, True}

LStringQ[s31, Alignment → Center, Trace → True]

{↓ ∧ ⇒ x_1 ⇔ y x1 z12 ⊕ ⇒ ∨ x x_3 x ↑ ¬ x_1 ¬ z}
{↓ ∧ ⇒ x$1 ⇔ y x1 z12 ⊕ ⇒ ∨ x x$3 x ↑ ¬ x$1 ¬ z}
{NOR, AND, IMPLIES, x_1, EQUIV, y, x1, z12, XOR, IMPLIES, OR, x, x_3, x, NAND, NOT, x_1, NOT, z}
{NOR, AND, IMPLIES, x_1, EQUIV, y, x1, z12, XOR, IMPLIES, OR, x, x_3, x, NAND, NOT, x_1, True}
{NOR, AND, IMPLIES, x_1, EQUIV, y, x1, z12, XOR, IMPLIES, OR, x, x_3, x, NAND, True, True}
{NOR, AND, IMPLIES, x_1, EQUIV, y, x1, z12, XOR, IMPLIES, OR, x, x_3, x, True}
{NOR, AND, IMPLIES, x_1, EQUIV, y, x1, z12, XOR, IMPLIES, True, x, True}
{NOR, AND, IMPLIES, x_1, EQUIV, y, x1, z12, XOR, True, True}
{NOR, AND, IMPLIES, x_1, EQUIV, y, x1, z12, True}
{NOR, AND, IMPLIES, x_1, True, z12, True}
{NOR, AND, True, z12, True}
{NOR, True, True}
{True}

Example 4

{s41 = "↓∧$x_1$$x_1$xy~NAND~↑x1x⇒y¬z", LStringQ[s41]}

{↓∧$x_1$$x_1$xy~NAND~↑x1x⇒y¬z, False}

LStringQ[s41, Trace → True];

Example 5

{s51 = "↓⇒a_1c⊕¬c¬b¬↑∨¬a¬b1∧¬a_1c", LStringQ[s51]}

{↓⇒a_1c⊕¬c¬b¬↑∨¬a¬b1∧¬a_1c, False}

LStringQ[s51, Alignment → Left, Trace -> True]

{↓ ⇒ a_1 c ⊕ ¬ c ¬ b ¬ ↑ ∨ ¬ a ¬ b1 ∧ ¬ a_1 c}
{↓ ⇒ a$1 c ⊕ ¬ c ¬ b ¬ ↑ ∨ ¬ a ¬ b1 ∧ ¬ a$1 c}
{NOR, IMPLIES, a_1, c, XOR, NOT, c, NOT, b, NOT, NAND, OR, NOT, a, NOT, b1, AND, NOT, a_1, c}
{NOR, IMPLIES, a_1, c, XOR, NOT, c, NOT, b, NOT, NAND, OR, NOT, a, NOT, b1, AND, True, c}
{NOR, IMPLIES, a_1, c, XOR, NOT, c, NOT, b, NOT, NAND, OR, NOT, a, NOT, b1, True}
{NOR, IMPLIES, a_1, c, XOR, NOT, c, NOT, b, NOT, NAND, OR, NOT, a, True, True}
{NOR, IMPLIES, a_1, c, XOR, NOT, c, NOT, b, NOT, NAND, OR, True, True, True}
{NOR, IMPLIES, a_1, c, XOR, NOT, c, NOT, b, NOT, NAND, True, True}
{NOR, IMPLIES, a_1, c, XOR, NOT, c, NOT, b, NOT, True}
{NOR, IMPLIES, a_1, c, XOR, NOT, c, NOT, b, True}
{NOR, IMPLIES, a_1, c, XOR, NOT, c, True, True}
{NOR, IMPLIES, a_1, c, XOR, True, True, True}
{NOR, IMPLIES, a_1, c, True, True}
{NOR, True, True, True}
{True, True}
{False}

Example 6

{s61 = "⇔yxz⊕∨⇒∧x12x_{32}↑xz¬x_1¬x¬yz", LStringQ[s61]}

{⇔yxz⊕∨⇒∧x12x_{32}↑xz¬x_1¬x¬yz, False}

```
LStringQ[s61, Alignment → Right, Trace → True]
```

```
{⇔ y x z ⊕ ∨ ⇒ ∧ x12 x_{32} ↑ x z ¬ x_{1} ¬ x ¬ y z}
{⇔ y x z ⊕ ∨ ⇒ ∧ x12 x$32 ↑ x z ¬ x$1 ¬ x ¬ y z}
{EQUIV, y, x, z, XOR, OR, IMPLIES, AND, x12, x_{32}, NAND, x, z, NOT, x_{1}, NOT, x, NOT, y, z}
{EQUIV, y, x, z, XOR, OR, IMPLIES, AND, x12, x_{32}, NAND, x, z, NOT, x_{1}, NOT, x, True, z}
{EQUIV, y, x, z, XOR, OR, IMPLIES, AND, x12, x_{32}, NAND, x, z, NOT, x_{1}, True, True, z}
{EQUIV, y, x, z, XOR, OR, IMPLIES, AND, x12, x_{32}, NAND, x, z, True, True, True, z}
{EQUIV, y, x, z, XOR, OR, IMPLIES, AND, x12, x_{32}, True, True, True, True, z}
{EQUIV, y, x, z, XOR, OR, IMPLIES, True, True, True, True, True, z}
{EQUIV, y, x, z, XOR, OR, True, True, True, True, z}
{EQUIV, y, x, z, XOR, True, True, True, z}
{EQUIV, y, x, z, True, True, z}
{True, z, True, True, z}
{False}
```

Conversion between Logical Strings and Logical Formulae

Example 1

```
{s11 = "↓∧xx1~NAND~x1x1¬y", LStringQ[s11], f11 = ToLFormula[s11]}
```

```
{↓∧xx1~NAND~x1x1¬y, False, $Failed}
```

```
{LStringQ[s11, Trace → True], ToLFormula[s11, Trace → True]} // Row[#, ", "] &
```

```
{↓ ∧ x x1 ↑ x1 x1 ¬ y}
{NOR, AND, x, x1, NAND, x1, x1, NOT, y}
{NOR, AND, x, x1, NAND, x1, x1, True}
{NOR, AND, x, x1, True, True}
{NOR, True, True, True}
{True, True}
{False}
,
{↓ ∧ x x1 ↑ x1 x1 ¬ y}
{NOR, AND, x, x1, NAND, x1, x1, NOT, y}
{NOR, AND, x, x1, NAND, x1, x1, ¬ y}
{NOR, AND, x, x1, x1↑x1, ¬ y}
{NOR, x∧x1, x1↑x1, ¬ y}
{(x∧x1) ↓ (x1↑x1), ¬ y}
{$Failed}
```

Example 2

```
{s21 = "↓∧xx_{1}~NAND~x1¬y", LStringQ[s21], f21 = ToLFormula[s21]}
```

```
{↓∧xx_{1}~NAND~x1¬y, True, (x∧x_{1}) ↓ (x1↑¬ y)}
```

```
ToLFormula[s21, Trace → True]
```

```
{↓ ∧ x x_{1} ↑ x1 ¬ y}
{NOR, AND, x, x_{1}, NAND, x1, NOT, y}
{NOR, AND, x, x_{1}, NAND, x1, ¬ y}
{NOR, AND, x, x_{1}, x1↑¬ y}
{NOR, x∧x_{1}, x1↑¬ y}
{(x∧x_{1}) ↓ (x1↑¬ y)}
```

```
ToLString[f21]
```

```
↓∧xx_{1}↑x1¬y
```

```
s22 = ToLString[f21]; f22 = ToLFormula[s22]; s23 = ToLString[f22];
{{s21, s22, s23}, {s21 === s22, s22 === s23, NormalizeLString /@ {s21, s23} // SameQ}} //
 Column[#, Center] &
```

```
{↓∧xx_{1}~NAND~x1¬y, ↓∧xx_{1}↑x1¬y, ↓∧xx_{1}↑x1¬y}
{False, True, True}
```

```
ToLString[f22, Trace → True]
```

```
{(x ∧ x_1) ↓ (x1 ↑ ¬ y)}
{(x ∧ x$1•) ↓ (x1 ↑ ¬ y)}
{NOR[AND[x, x$1•], NAND[x1, NOT[y]]]}
{NOR AND x x$1• NAND x1 NOT y}
{↓, ∧, x, x$1•, ↑, x1, ¬, y}
{↓∧xx$1•↑x1¬y}
{↓∧xx_1↑x1¬y}
```

Example 3

```
{s31 = "↓∧⇒x_1⇔yx1z12⊕⇒∨xx_3x↑¬x_1¬z", LStringQ[s31], f31 = ToLFormula[s31]}
```

```
{↓∧⇒x_1⇔yx1z12⊕⇒∨xx_3x↑¬x_1¬z, True, ((x_1 ⇒ (y ⇔ x1)) ∧ z12) ↓ (((x ∨ x_3) ⇒ x) ⊕ (¬ x_1 ↑ ¬ z))}
```

```
ToLFormula[s31, Trace → True]
```

```
{↓ ∧ ⇒ x_1 ⇔ y x1 z12 ⊕ ⇒ ∨ x x_3 x ↑ ¬ x_1 ¬ z}
{NOR, AND, IMPLIES, x_1, EQUIV, y, x1, z12, XOR, IMPLIES, OR, x, x_3, x, NAND, NOT, x_1, NOT, z}
{NOR, AND, IMPLIES, x_1, EQUIV, y, x1, z12, XOR, IMPLIES, OR, x, x_3, x, NAND, NOT, x_1, ¬ z}
{NOR, AND, IMPLIES, x_1, EQUIV, y, x1, z12, XOR, IMPLIES, OR, x, x_3, x, NAND, ¬ x_1, ¬ z}
{NOR, AND, IMPLIES, x_1, EQUIV, y, x1, z12, XOR, IMPLIES, OR, x, x_3, x, ¬ x_1 ↑ ¬ z}
{NOR, AND, IMPLIES, x_1, EQUIV, y, x1, z12, XOR, IMPLIES, x ∨ x_3, x, ¬ x_1 ↑ ¬ z}
{NOR, AND, IMPLIES, x_1, EQUIV, y, x1, z12, XOR, (x ∨ x_3) ⇒ x, ¬ x_1 ↑ ¬ z}
{NOR, AND, IMPLIES, x_1, EQUIV, y, x1, z12, ((x ∨ x_3) ⇒ x) ⊕ (¬ x_1 ↑ ¬ z)}
{NOR, AND, IMPLIES, x_1, y ⇔ x1, z12, ((x ∨ x_3) ⇒ x) ⊕ (¬ x_1 ↑ ¬ z)}
{NOR, AND, x_1 ⇒ (y ⇔ x1), z12, ((x ∨ x_3) ⇒ x) ⊕ (¬ x_1 ↑ ¬ z)}
{NOR, (x_1 ⇒ (y ⇔ x1)) ∧ z12, ((x ∨ x_3) ⇒ x) ⊕ (¬ x_1 ↑ ¬ z)}
{((x_1 ⇒ (y ⇔ x1)) ∧ z12) ↓ (((x ∨ x_3) ⇒ x) ⊕ (¬ x_1 ↑ ¬ z))}
```

```
s32 = ToLString[f31]; f32 = ToLFormula[s32]; s33 = ToLString[f32];
{s31, s32, s33, {s31 === s32, s32 === s33, NormalizeLString /@ {s31, s33} // SameQ}} //
 Column[#, Center] &
```

```
↓∧⇒x_1⇔yx1z12⊕⇒∨xx_3x↑¬x_1¬z
↓∧⇒x_1⇔yx1z12⊕⇒∨xx_3x↑¬x_1¬z
↓∧⇒x_1⇔yx1z12⊕⇒∨xx_3x↑¬x_1¬z
{False, True, True}
```

```
ToLString[f32, Trace → True]
```

```
{((x_1 ⇒ (y ⇔ x1)) ∧ z12) ↓ (((x ∨ x_3) ⇒ x) ⊕ (¬ x_1 ↑ ¬ z))}
{((x$1• ⇒ (y ⇔ x1)) ∧ z12) ↓ (((x ∨ x$3•) ⇒ x) ⊕ (¬ x$1• ↑ ¬ z))}
{NOR[AND[IMPLIES[x$1•, EQUIV[y, x1]],
  z12], XOR[IMPLIES[OR[x, x$3•], x], NAND[NOT[x$1•], NOT[z]]]]}
{NOR AND IMPLIES x$1• EQUIV y x1 z12 XOR IMPLIES OR x x$3• x NAND NOT x$1• NOT z}
{↓, ∧, ⇒, x$1•, ⇔, y, x1, z12, ⊕, ⇒, ∨, x, x$3•, x, ↑, ¬, x$1•, ¬, z}
{↓∧⇒x$1•⇔yx1z12⊕⇒∨xx$3•x↑¬x$1•¬z}
{↓∧⇒x_1⇔yx1z12⊕⇒∨xx_3x↑¬x_1¬z}
```

Example 4

```
{s41 = "↓∧x_1x_1xy~NAND~↑x1x⇒y¬z", LStringQ[s41], ToLFormula[s41]}
```

```
{↓∧x_1x_1xy~NAND~↑x1x⇒y¬z, False, $Failed}
```

ToLFormula[s41, Alignment → Left, Trace → True]

{↓ ∧ x_1 x_1 x y ↑ ↑ x1 x ⇒ y ¬ z}
{NOR, AND, x_1, x_1, x, y, NAND, NAND, x1, x, IMPLIES, y, NOT, z}
{NOR, AND, x_1, x_1, x, y, NAND, NAND, x1, x, IMPLIES, y, ¬ z}
{NOR, AND, x_1, x_1, x, y, NAND, NAND, x1, x, y ⇒ ¬ z}
{NOR, AND, x_1, x_1, x, y, NAND, x1 ↑ x, y ⇒ ¬ z}
{NOR, AND, x_1, x_1, x, y, (x1 ↑ x) ↑ (y ⇒ ¬ z)}
{NOR, x_1 ∧ x_1, x, y, (x1 ↑ x) ↑ (y ⇒ ¬ z)}
{(x_1 ∧ x_1) ↓ x, y, (x1 ↑ x) ↑ (y ⇒ ¬ z)}
{$Failed}

Example 5

{s51 = "↓⇒a_1c⊕¬c¬b¬↑∨¬a¬b1∧¬a_1c", LStringQ[s51], ToLFormula[s51]}

{↓⇒a_1c⊕¬c¬b¬↑∨¬a¬b1∧¬a_1c, False, $Failed}

ToLFormula[s51, Alignment → Right, Trace → True]

{↓ ⇒ a_1 c ⊕ ¬ c ¬ b ¬ ↑ ∨ ¬ a ¬ b1 ∧ ¬ a_1 c}
{NOR, IMPLIES, a_1, c, XOR, NOT, c, NOT, b, NOT, NAND, OR, NOT, a, NOT, b1, AND, NOT, a_1, c}
{NOR, IMPLIES, a_1, c, XOR, NOT, c, NOT, b, NOT, NAND, OR, NOT, a, NOT, b1, AND, ¬ a_1, c}
{NOR, IMPLIES, a_1, c, XOR, NOT, c, NOT, b, NOT, NAND, OR, NOT, a, NOT, b1, ¬ a_1 ∧ c}
{NOR, IMPLIES, a_1, c, XOR, NOT, c, NOT, b, NOT, NAND, OR, NOT, a, ¬ b1, ¬ a_1 ∧ c}
{NOR, IMPLIES, a_1, c, XOR, NOT, c, NOT, b, NOT, NAND, OR, ¬ a, ¬ b1, ¬ a_1 ∧ c}
{NOR, IMPLIES, a_1, c, XOR, NOT, c, NOT, b, NOT, NAND, ¬ a ∨ ¬ b1, ¬ a_1 ∧ c}
{NOR, IMPLIES, a_1, c, XOR, NOT, c, NOT, b, NOT, (¬ a ∨ ¬ b1) ↑ (¬ a_1 ∧ c)}
{NOR, IMPLIES, a_1, c, XOR, NOT, c, NOT, b, ¬ ((¬ a ∨ ¬ b1) ↑ (¬ a_1 ∧ c))}
{NOR, IMPLIES, a_1, c, XOR, NOT, c, ¬ b, ¬ ((¬ a ∨ ¬ b1) ↑ (¬ a_1 ∧ c))}
{NOR, IMPLIES, a_1, c, XOR, ¬ c, ¬ b, ¬ ((¬ a ∨ ¬ b1) ↑ (¬ a_1 ∧ c))}
{NOR, IMPLIES, a_1, c, ¬ c ⊕ ¬ b, ¬ ((¬ a ∨ ¬ b1) ↑ (¬ a_1 ∧ c))}
{NOR, a_1 ⇒ c, ¬ c ⊕ ¬ b, ¬ ((¬ a ∨ ¬ b1) ↑ (¬ a_1 ∧ c))}
{(a_1 ⇒ c) ↓ (¬ c ⊕ ¬ b), ¬ ((¬ a ∨ ¬ b1) ↑ (¬ a_1 ∧ c))}
{$Failed}

Example 6

{s61 = "⇔yxz⊕∨⇒∧x12x_{32}↑xz¬x_1¬x¬yz", LStringQ[s61], ToLFormula[s61]}

{⇔yxz⊕∨⇒∧x12x_{32}↑xz¬x_1¬x¬yz, False, $Failed}

ToLFormula[s61, Trace → True]

{⇔ y x z ⊕ ∨ ⇒ ∧ x12 x_{32} ↑ x z ¬ x_1 ¬ x ¬ y z}
{EQUIV, y, x, z, XOR, OR, IMPLIES, AND, x12, x_{32}, NAND, x, z, NOT, x_1, NOT, x, NOT, y, z}
{EQUIV, y, x, z, XOR, OR, IMPLIES, AND, x12, x_{32}, NAND, x, z, NOT, x_1, NOT, x, ¬ y, z}
{EQUIV, y, x, z, XOR, OR, IMPLIES, AND, x12, x_{32}, NAND, x, z, NOT, x_1, ¬ x, ¬ y, z}
{EQUIV, y, x, z, XOR, OR, IMPLIES, AND, x12, x_{32}, NAND, x, z, ¬ x_1, ¬ x, ¬ y, z}
{EQUIV, y, x, z, XOR, OR, IMPLIES, AND, x12, x_{32}, x ↑ z, ¬ x_1, ¬ x, ¬ y, z}
{EQUIV, y, x, z, XOR, OR, IMPLIES, x12 ∧ x_{32}, x ↑ z, ¬ x_1, ¬ x, ¬ y, z}
{EQUIV, y, x, z, XOR, OR, (x12 ∧ x_{32}) ⇒ (x ↑ z), ¬ x_1, ¬ x, ¬ y, z}
{EQUIV, y, x, z, XOR, ((x12 ∧ x_{32}) ⇒ (x ↑ z)) ∨ ¬ x_1, ¬ x, ¬ y, z}
{EQUIV, y, x, z, (((x12 ∧ x_{32}) ⇒ (x ↑ z)) ∨ ¬ x_1) ⊕ ¬ x, ¬ y, z}
{y ⇔ x, z, (((x12 ∧ x_{32}) ⇒ (x ↑ z)) ∨ ¬ x_1) ⊕ ¬ x, ¬ y, z}
{$Failed}

TruthTable

TruthTable[x, options] computes the truth table of the logical formula or list of logical formulae x and outputs or prints it as Grid.

Alternatives for options (default value = the first alternative) :
- BooleanValues -> { 0, 1} | {False,True} | -> {FalseSymbol, TrueSymbol},
- ItemSize -> Automatic | Full |as for Grid,
- Labels -> None | Automatic | list of sufficient length,
- Print -> False | True,
- ReverseValues -> False | True,
- SelectValuations -> All | AllTrue | Mixed | AllFalse | LastTrue | LastFalse | OnlyLastFalse |
{{ (indexes | labels of) formulae} -> FalseSymbol, {(indexes | labels of) formulae} -> TrueSymbol} |
{{ (indexes | labels of) formulae} -> TrueSymbol, {(indexes | labels of) formulae} -> FalseSymbol},
- TableBreaks -> None | integer > 1 | increasing list of integers > 1,
- Transpose -> False | True,
- Variables -> All | List of logical variables.
ItemSize is an option of Grid and Transpose is an option of List, Matrix and Tensor.

```
Options[TruthTable]
```

```
{BooleanValues → {0, 1}, ItemSize → Automatic, Labels → None,
 Print → False, ReverseValues → False, SelectValuations → All,
 TableBreaks → None, Transpose → False, Variables → All}
```

BooleanValues, ReverseValues, Transpose

Example 1

```
α = ¬ q ⇒ (r↓¬ q);
β = q⊕¬ r;
γ = (q∧r) ⇔ (¬ q⇒r);
M1 = HList[α, β, γ];
{TruthTable[M1], ",   ", TruthTable[ReleaseHE[M1], BooleanValues → {𝔽, 𝕋}]} // Row
```

q	r	α	β	γ
0	0	0	1	1
0	1	0	0	0
1	0	1	0	0
1	1	1	1	1

,

q	r	¬ q ⇒ r↓(¬ q)	q⊕(¬ r)	q∧r ⇔ (¬ q ⇒ r)
𝔽	𝔽	𝔽	𝕋	𝕋
𝔽	𝕋	𝔽	𝔽	𝔽
𝕋	𝔽	𝕋	𝔽	𝔽
𝕋	𝕋	𝕋	𝕋	𝕋

Example 2

```
M2 = {α, β, γ} /. r → r1;
{TruthTable[M2, ReverseValues → True], " ,   ",
  TruthTable[M2, ReverseValues → True, Transpose → True]} // Row
```

q	r_1	¬ q ⇒ r_1↓(¬ q)	q⊕(¬ r_1)	q∧r_1 ⇔ (¬ q ⇒ r_1)
1	1	1	1	1
1	0	1	0	0
0	1	0	0	0
0	0	0	1	1

,

q	1	1	0	0
r_1	1	0	1	0
¬ q ⇒ r_1↓(¬ q)	1	1	0	0
q⊕(¬ r_1)	1	0	0	1
q∧r_1 ⇔ (¬ q ⇒ r_1)	1	0	0	1

Example 3

```
M3 = {α, β, γ};
TruthTable[M3, BooleanValues → #, Transpose → True] & /@ {{"", 𝕋}, {𝔽, ""}} // Row[#, ", "] &
```

q			𝕋	𝕋
r		𝕋		𝕋
¬ q ⇒ r ↓ (¬ q)			𝕋	𝕋
q ⊕ (¬ r)	𝕋			𝕋
q ∧ r ⇔ (¬ q ⇒ r)	𝕋			𝕋

,

q	𝔽	𝔽		
r	𝔽		𝔽	
¬ q ⇒ r ↓ (¬ q)	𝔽	𝔽		
q ⊕ (¬ r)		𝔽	𝔽	
q ∧ r ⇔ (¬ q ⇒ r)		𝔽	𝔽	

Labels

Example 4

```
M3 = {α, β, γ};
{TruthTable[M3, Labels → Automatic], TruthTable[M3, Labels → HList[α, β, γ]]} // Column
```

•	•	1	2	3
q	r	¬ q ⇒ r ↓ (¬ q)	q ⊕ (¬ r)	q ∧ r ⇔ (¬ q ⇒ r)
0	0	0	1	1
0	1	0	0	0
1	0	1	0	0
1	1	1	1	1

•	•	α	β	γ
q	r	¬ q ⇒ r ↓ (¬ q)	q ⊕ (¬ r)	q ∧ r ⇔ (¬ q ⇒ r)
0	0	0	1	1
0	1	0	0	0
1	0	1	0	0
1	1	1	1	1

Example 5

```
M3 = {α, β, γ};
{TruthTable[M3, Labels → {"α", "β", "γ"}, Transpose → True], " , ",
  TruthTable[M3, Labels → CharacterRange["α", "γ"], Transpose → True]} // Row
```

•	q	0	0	1	1
•	r	0	1	0	1
α	¬ q ⇒ r ↓ (¬ q)	0	0	1	1
β	q ⊕ (¬ r)	1	0	0	1
γ	q ∧ r ⇔ (¬ q ⇒ r)	1	0	0	1

,

•	q	0	0	1	1
•	r	0	1	0	1
α	¬ q ⇒ r ↓ (¬ q)	0	0	1	1
β	q ⊕ (¬ r)	1	0	0	1
γ	q ∧ r ⇔ (¬ q ⇒ r)	1	0	0	1

■ SelectValuations

Example 6

```
α = ¬ q ⇒ (r ↓ ¬ q);
β = q ⊕ ¬ r;
γ = (q ∧ r) ⇔ (¬ q ⇒ r);
δ = (¬ r ⇒ s) ↑ (s ∨ ¬ r);
ε = (q ∧ r) ∨ s;
M4 = {α, β, γ, δ, ε};
TruthTable[M4]
```

q	r	s	¬ q ⇒ r↓ (¬ q)	q ⊕ (¬ r)	q ∧ r ⇔ (¬ q ⇒ r)	(¬ r ⇒ s) ↑ (s ∨ ¬ r)	(q ∧ r) ∨ s
0	0	0	0	1	1	1	0
0	0	1	0	1	1	0	1
0	1	0	0	0	0	1	0
0	1	1	0	0	0	0	1
1	0	0	1	0	0	1	0
1	0	1	1	0	0	0	1
1	1	0	1	1	1	1	1
1	1	1	1	1	1	0	1

```
M4 = {α, β, γ, δ, ε};
TruthTable[M4, SelectValuations → AllTrue]
```

q	r	s	¬ q ⇒ r↓ (¬ q)	q ⊕ (¬ r)	q ∧ r ⇔ (¬ q ⇒ r)	(¬ r ⇒ s) ↑ (s ∨ ¬ r)	(q ∧ r) ∨ s
1	1	0	1	1	1	1	1

```
M4 = {α, β, γ, δ, ε};
TruthTable[M4, SelectValuations → Mixed]
```

q	r	s	¬ q ⇒ r↓ (¬ q)	q ⊕ (¬ r)	q ∧ r ⇔ (¬ q ⇒ r)	(¬ r ⇒ s) ↑ (s ∨ ¬ r)	(q ∧ r) ∨ s
0	0	0	0	1	1	1	0
0	0	1	0	1	1	0	1
0	1	0	0	0	0	1	0
0	1	1	0	0	0	0	1
1	0	0	1	0	0	1	0
1	0	1	1	0	0	0	1
1	1	1	1	1	1	0	1

```
M4 = {α, β, γ, δ, ε};
TruthTable[M4, SelectValuations → AllFalse]
```

No valuations have been selected

```
M4 = {α, β, γ, δ, ε};
TruthTable[M4, SelectValuations → LastTrue]
```

q	r	s	¬ q ⇒ r↓ (¬ q)	q ⊕ (¬ r)	q ∧ r ⇔ (¬ q ⇒ r)	(¬ r ⇒ s) ↑ (s ∨ ¬ r)	(q ∧ r) ∨ s
0	0	1	0	1	1	0	1
0	1	1	0	0	0	0	1
1	0	1	1	0	0	0	1
1	1	0	1	1	1	1	1
1	1	1	1	1	1	0	1

```
M4 = {α, β, γ, δ, ε};
TruthTable[M4, SelectValuations → LastFalse]
```

q	r	s	¬ q ⇒ r↓ (¬ q)	q ⊕ (¬ r)	q ∧ r ⇔ (¬ q ⇒ r)	(¬ r ⇒ s) ↑ (s ∨ ¬ r)	(q ∧ r) ∨ s
0	0	0	0	1	1	1	0
0	1	0	0	0	0	1	0
1	0	0	1	0	0	1	0

```
M = {α, β, γ, δ, ε};
TruthTable[M, SelectValuations → OnlyLastFalse]
```

No valuations have been selected

Example 7

```
M4 = {α, β, γ, δ, ε};
TruthTable[M4, Labels → Automatic, SelectValuations → {{3} → 0}]
```

•	•	•	1	2	3	4	5
q	r	s	¬ q ⇒ r ↓ (¬ q)	q ⊕ (¬ r)	q ∧ r ⇔ (¬ q ⇒ r)	(¬ r ⇒ s) ↑ (s ∨ ¬ r)	(q ∧ r) ∨ s
0	1	0	0	0	0	1	0
0	1	1	0	0	0	0	1
1	0	0	1	0	0	1	0
1	0	1	1	0	0	0	1

```
M4 = {α, β, γ, δ, ε};
TruthTable[M4, Labels → CharacterRange["α", "ε"], SelectValuations → #] & /@
  {{{3, 5, "β"} → 0, {1, 4} → 1}, {{1, 4} → 1, {3, 5, "β"} → 0}} // Column
```

•	•	•	α	β	γ	δ	ε
q	r	s	¬ q ⇒ r ↓ (¬ q)	q ⊕ (¬ r)	q ∧ r ⇔ (¬ q ⇒ r)	(¬ r ⇒ s) ↑ (s ∨ ¬ r)	(q ∧ r) ∨ s
1	0	0	1	0	0	1	0

•	•	•	α	β	γ	δ	ε
q	r	s	¬ q ⇒ r ↓ (¬ q)	q ⊕ (¬ r)	q ∧ r ⇔ (¬ q ⇒ r)	(¬ r ⇒ s) ↑ (s ∨ ¬ r)	(q ∧ r) ∨ s
1	0	0	1	0	0	1	0

```
M4 = {α, β, γ, δ, ε};
TruthTable[M, BooleanValues → {𝔽, 𝕋}, Labels → HList[α, β, γ, δ, ε],
 SelectValuations → {{HoldForm[α], 4} → 𝕋, {5} → 𝔽}]
```

•	•	•	α	β	γ	δ	ε
q	r	s	¬ q ⇒ r ↓ (¬ q)	q ⊕ (¬ r)	q ∧ r ⇔ (¬ q ⇒ r)	(¬ r ⇒ s) ↑ (s ∨ ¬ r)	(q ∧ r) ∨ s
𝕋	𝔽	𝔽	𝕋	𝔽	𝔽	𝕋	𝔽

■ TableBreaks and Print

Example 8

```
M4 = {α, β, γ, δ, ε};
TruthTable[M4, BooleanValues → {0, 1}, Labels → {"α", "β", "γ", "δ", "ε"},
  ReverseValues → False, TableBreaks → 3, Transpose → True] // Row[#, ",   "] &
```

•	q	0	0	0
•	r	0	0	1
•	s	0	1	0
α	¬ q ⇒ r↓ (¬ q)	0	0	0
β	q ⊕ (¬ r)	1	1	0
γ	q ∧ r ⇔ (¬ q ⇒ r)	1	1	0
δ	(¬ r ⇒ s) ↑ (s ∨ ¬ r)	1	0	1
ε	(q ∧ r) ∨ s	0	1	0

,

•	q	0	1	1
•	r	1	0	0
•	s	1	0	1
α	¬ q ⇒ r↓ (¬ q)	0	1	1
β	q ⊕ (¬ r)	0	0	0
γ	q ∧ r ⇔ (¬ q ⇒ r)	0	0	0
δ	(¬ r ⇒ s) ↑ (s ∨ ¬ r)	0	1	0
ε	(q ∧ r) ∨ s	1	0	1

,

•	q
•	r
•	s
α	¬ q ⇒ r↓
β	q ⊕ (¬
γ	q ∧ r ⇔ (¬
δ	(¬ r ⇒ s) ↑
ε	(q ∧ r)

Example 9

```
M4 = {α, β, γ, δ, ε};
TruthTable[M4, BooleanValues → {0, 1}, Labels → None(*{"α","β","γ","δ","ε"}*),
 Print → True, ReverseValues → False, TableBreaks → {4, 6}, Transpose → False]
```

q	r	s	¬ q ⇒ r↓ (¬ q)	q ⊕ (¬ r)	q ∧ r ⇔ (¬ q ⇒ r)	(¬ r ⇒ s) ↑ (s ∨ ¬ r)	(q ∧ r) ∨ s
0	0	0	0	1	1	1	0
0	0	1	0	1	1	0	1
0	1	0	0	0	0	1	0
0	1	1	0	0	0	0	1

q	r	s	¬ q ⇒ r↓ (¬ q)	q ⊕ (¬ r)	q ∧ r ⇔ (¬ q ⇒ r)	(¬ r ⇒ s) ↑ (s ∨ ¬ r)	(q ∧ r) ∨ s
1	0	0	1	0	0	1	0
1	0	1	1	0	0	0	1

q	r	s	¬ q ⇒ r↓ (¬ q)	q ⊕ (¬ r)	q ∧ r ⇔ (¬ q ⇒ r)	(¬ r ⇒ s) ↑ (s ∨ ¬ r)	(q ∧ r) ∨ s
1	1	0	1	1	1	1	1
1	1	1	1	1	1	0	1

Example 10

```
M4 = {α, β, γ, δ, ε};
TruthTable[M4, BooleanValues → {0, 1}, Labels → {"α", "β", "γ", "δ", "ε"}, Print → False,
  ReverseValues → False, SelectValuations → All, TableBreaks → 3, Transpose → False] // Column
```

•	•	•	α	β	γ	δ	ε
q	r	s	¬ q ⇒ r↓ (¬ q)	q ⊕ (¬ r)	q ∧ r ⇔ (¬ q ⇒ r)	(¬ r ⇒ s) ↑ (s ∨ ¬ r)	(q ∧ r) ∨ s
0	0	0	0	1	1	1	0
0	0	1	0	1	1	0	1
0	1	0	0	0	0	1	0

•	•	•	α	β	γ	δ	ε
q	r	s	¬ q ⇒ r↓ (¬ q)	q ⊕ (¬ r)	q ∧ r ⇔ (¬ q ⇒ r)	(¬ r ⇒ s) ↑ (s ∨ ¬ r)	(q ∧ r) ∨ s
0	1	1	0	0	0	0	1
1	0	0	1	0	0	1	0
1	0	1	1	0	0	0	1

•	•	•	α	β	γ	δ	ε
q	r	s	¬ q ⇒ r↓ (¬ q)	q ⊕ (¬ r)	q ∧ r ⇔ (¬ q ⇒ r)	(¬ r ⇒ s) ↑ (s ∨ ¬ r)	(q ∧ r) ∨ s
1	1	0	1	1	1	1	1
1	1	1	1	1	1	0	1

Example 11

```
M4 = {α, β, γ, δ, ε};
TruthTable[M, Labels → {"α", "β", "γ", "δ", "ε"},
 StylePrint → True, Variables → {q, s}, Print → True]
```

•	•	α	β	γ	δ	ε
q	s	¬ q ⇒ r↓ (¬ q)	q ⊕ (¬ r)	q ∧ r ⇔ (¬ q ⇒ r)	(¬ r ⇒ s) ↑ (s ∨ ¬ r)	(q ∧ r) ∨ s
0	0	0	¬ r	0 ⇔ (1 ⇒ r)	(¬ r ⇒ 0) ↑ (¬ r)	0
0	1	0	¬ r	0 ⇔ (1 ⇒ r)	0	1
1	0	1	r	r ⇔ 1	(¬ r ⇒ 0) ↑ (¬ r)	r
1	1	1	r	r ⇔ 1	0	1

EmptyTruthTable

EmptyTruthTable[x, options] is a version of TruthTable with no truth-values for formulas.
Alternatives for options (default value = the first alternative) :
- BooleanValues -> { 0, 1} | {False,True} | -> {FalseSymbol, TrueSymbol},
- ItemSize -> Automatic | Full | as for Grid,
- Labels -> None | Automatic | list of sufficient length,
- Print -> False | True,
- ReverseValues -> False | True,
- TableBreaks -> None | integer > 1 | increasing list of integers > 1,
- Transpose -> False | True.
ItemSize is an option of Grid and Transpose is an option of List, Matrix and Tensor.

```
Options[EmptyTruthTable]
```

```
{BooleanValues → {0, 1}, ItemSize → Automatic, Labels → None,
 Print → False, ReverseValues → False, SelectValuations → All,
 TableBreaks → None, Transpose → False, Variables → All}
```

Example 1

```
α = ¬ q ⇒ (r↓¬ q);
β = q⊕¬ r;
γ = (q∧r) ⇔ (¬ q ⇒ r);
M = {α, β, γ};
{EmptyTruthTable[M], EmptyTruthTable[M, BooleanValues → {𝔽, 𝕋}, ReverseValues → True,
   ItemSize → {{1, 2, {Automatic}}, {1.5, {2}}}]} // Column[#, Center] &
```

q	r	¬ q ⇒ r↓ (¬ q)	q⊕ (¬ r)	q∧r ⇔ (¬ q ⇒ r)
0	0			
0	1			
1	0			
1	1			

q	r	¬ q ⇒ r↓ (¬ q)	q⊕ (¬ r)	q∧r ⇔ (¬ q ⇒ r)
𝕋	𝕋			
𝕋	𝔽			
𝔽	𝕋			
𝔽	𝔽			

Example 2

```
M = {α, β, γ};
EmptyTruthTable[M, ItemSize → Full, Labels → Automatic, Print → True, TableBreaks → None];
EmptyTruthTable[M, BooleanValues → {𝔽, 𝕋}, ItemSize → Full,
 Labels → {"α", "β", "γ"}, TableBreaks → {1, 3}, Transpose → True, Print → False]
```

•	•	1	2	3
q	r	¬ q ⇒ r↓ (¬ q)	q⊕ (¬ r)	q∧r ⇔ (¬ q ⇒ r)
0	0			
0	1			
1	0			
1	1			

{

•	q	𝔽
•	r	𝔽
α	¬ q ⇒ r↓ (¬ q)	
β	q⊕ (¬ r)	
γ	q∧r ⇔ (¬ q ⇒ r)	

,

•	q	𝔽	𝕋
•	r	𝕋	𝔽
α	¬ q ⇒ r↓ (¬ q)		
β	q⊕ (¬ r)		
γ	q∧r ⇔ (¬ q ⇒ r)		

,

•	q	𝕋
•	r	𝕋
α	¬ q ⇒ r↓ (¬ q)	
β	q⊕ (¬ r)	
γ	q∧r ⇔ (¬ q ⇒ r)	

}

TautologyQ and LEquivalentQ

LTautologyQ[*x*] returns True if *x* is a tautology, and False if *x* is a logical formula but not a tautology. LTautologyQ has the attribut Listable.

```
τ1 = (a ∨ (b ∧ c) ⇔ (a ∨ b) ∧ (a ∨ c));
τ2 = ((a ⇒ b) ∧ (b ⇒ c)) ⇒ (a ⇒ c);
τ3 = (a ⇒ b) ⇒ ((b ⇒ c) ⇒ (a ⇒ c));
τ4 = ¬ a ⇒ (¬ b ⇔ (b ⇒ a));
{τ1, τ2, τ3, LTautologyQ[{τ1, τ2, τ3, τ4}]} // Column[#, Center] &
```

```
a ∨ (b ∧ c) ⇔ (a ∨ b) ∧ (a ∨ c)
(a ⇒ b) ∧ (b ⇒ c) ⇒ (a ⇒ c)
(a ⇒ b) ⇒ ((b ⇒ c) ⇒ (a ⇒ c))
{True, True, True, True}
```

LEquivalentQ[*x*, *y*] returns True if *x*, *y* are logically equivalent logical formulas, and False in the opposite case.

```
{σ1, τ1} = {a ∨ (b ∧ c), (a ∨ b) ∧ (a ∨ c)};
{σ2, τ2} = {(¬ a ⇒ b), (a ⇒ b) ⇒ b};
{σ3, τ3} = {(a ⇒ (¬ b ∨ c)) ∧ ((c ↑ b) ⇒ (a ∨ ¬ b)), c ∨ ¬ b};
LEquivalentQ @@ # & /@ {{σ1, τ1}, {σ2, τ2}, {σ3, τ3}}
```

```
{True, True, True}
```

Normal Forms

CNF (Conjunctive Normal Form)

CNF[x, Complete | 1] returns the complete CNF of a logical formula x obtained from the truth table of x.
CNF[x, Minimal | 0] returns a simplified DNF of x obtained by the system function BooleanMinimize.
The second argument is optional and its default value is Simplified | 0.

```
{α, β, γ, δ, ε} = {a ⇔ (¬ b ↓ d), a ⇒ (c ↓ d), (((a ↑ b) ⊕ c) ↓ d) ↑ e, (α ↓ d) ∧ (b ⇒ ¬ a), α ⇒ γ};
M3 = {α, β, γ, δ, ε};
M3 // ReleaseHE // Column[#, Center] &
```

```
a ⇔ (¬ b) ↓ d
a ⇒ c ↓ d
((a ↑ b ⊕ c) ↓ d) ↑ e
(a ⇔ (¬ b) ↓ d) ↓ d ∧ (b ⇒ ¬ a)
a ⇔ (¬ b) ↓ d ⇒ ((a ↑ b ⊕ c) ↓ d) ↑ e
```

```
{CNF[α, 0], CNF[α, 1]} // Column[#, Center] &
```

```
(b ∨ ¬ a) ∧ (¬ a ∨ ¬ d) ∧ (a ∨ d ∨ ¬ b)
(a ∨ d ∨ ¬ b) ∧ (b ∨ d ∨ ¬ a) ∧ (b ∨ ¬ a ∨ ¬ d) ∧ (¬ a ∨ ¬ b ∨ ¬ d)
```

```
CNF[M3, 0] // Column[#, Center] &
```

```
(b ∨ ¬ a) ∧ (¬ a ∨ ¬ d) ∧ (a ∨ d ∨ ¬ b)
(¬ a ∨ ¬ c) ∧ (¬ a ∨ ¬ d)
(a ∨ d ∨ ¬ c ∨ ¬ e) ∧ (b ∨ d ∨ ¬ c ∨ ¬ e) ∧ (c ∨ d ∨ ¬ a ∨ ¬ b ∨ ¬ e)
¬ d ∧ (a ∨ b) ∧ (¬ a ∨ ¬ b)
(a ∨ b ∨ d ∨ ¬ c ∨ ¬ e) ∧ (c ∨ d ∨ ¬ a ∨ ¬ b ∨ ¬ e)
```

DNF (Disjunctive Normal Form)

DNF[x, Complete | 1] returns the complete DNF of a logical formula x obtained from the truth table of x.
DNF[x, Minimal | 0] returns a simplified DNF of x obtained by the system function BooleanMinimize.
The second argument is optional and its default value is Simplified | 0.

```
{α, β, γ, δ, ε} = {a ⇔ (¬ b↓d), a ⇒ (c↓d), (((a↑b) ⊕c) ↓d) ↑e, (α↓d) ∧ (b ⇒ ¬ a), α ⇒ γ};
M3 = {α, β, γ, δ, ε};
M3 // ReleaseHE // Column[#, Center] &
```

a ⇔ (¬ b) ↓d
a ⇒ c↓d
((a↑b⊕c) ↓d) ↑e
(a ⇔ (¬ b) ↓d) ↓d∧ (b ⇒ ¬ a)
a ⇔ (¬ b) ↓d ⇒ ((a↑b⊕c) ↓d) ↑e

```
{DNF[α, 0], DNF[α, 1]} // Column[#, Center] &
```

(d∧¬ a) ∨ (¬ a∧¬ b) ∨ (a∧b∧¬ d)
(a∧b∧¬ d) ∨ (b∧d∧¬ a) ∨ (d∧¬ a∧¬ b) ∨ (¬ a∧¬ b∧¬ d)

```
DNF[M3, 0] // Column[#, Center] &
```

(d∧¬ a) ∨ (¬ a∧¬ b) ∨ (a∧b∧¬ d)
(¬ c∧¬ d) ∨ ¬ a
d∨ (¬ a∧¬ c) ∨ (¬ b∧¬ c) ∨ (a∧b∧c) ∨ ¬ e
(a∧¬ b∧¬ d) ∨ (b∧¬ a∧¬ d)
d∨ (a∧¬ b) ∨ (b∧c) ∨ (b∧¬ a) ∨ (¬ b∧¬ c) ∨ ¬ e

Conversion to Complete Sets of Connectives: {¬ , ∧}, {¬ , ∨}, {¬ , ⇒}, {↑}, {↓}, {¬ , ∧ , ⊕}

ConvertFormula[*x*, form] uses system function BooleanConvert and finds a formula *y* logically equivalent to *x* and containing only connectives determined by the argument form:
- if form is “AND”, formula y contains only connectives AND and NOT;
- if form is “OR”, formula y contains only connectives OR and NOT;
- if form is “IMPLIES”, formula y contains only connectives IMPLIES and NOT;
- if form is “NAND”, formula y contains only connectives NAND and NOT;
- if form is “NOR”, formula y contains only connectives NOR and NOT;
- if form is “XOR”, formula y contains only connectives XOR, AND and NOT.

```
α = (a ⇔ ¬ (b↓d)) ⇒ ((a↑¬ b⊕c) ∨d) ∧e;
```

```
ConvertFormula[α, {"AND", "OR", "IMPLIES", "NAND", "NOR", "XOR"}] //
 Column[#, Center, Dividers → All, Spacings → 1] &
```

¬ (a∧b∧¬ e) ∧¬ (a∧d∧¬ e) ∧¬ (a∧b∧c∧¬ d) ∧¬ (¬ a∧¬ b∧¬ d∧¬ e)
¬ (a∨¬ b) ∨¬ (a∨¬ d) ∨¬ (b∨¬ e) ∨¬ (c∨¬ e) ∨¬ (¬ d∨¬ e) ∨¬ (b∨d∨¬ a)
(a ⇒ ¬ ((b ⇒ ¬ ((d ⇒ ¬ e) ⇒ ¬ (¬ d ⇒ (e ⇒ c)))) ⇒ ¬ (¬ b ⇒ ¬ (d ⇒ e)))) ⇒ ¬ (¬ a ⇒ ¬ (¬ b ⇒ (¬ d ⇒ e)))
(a↑ (¬ b) ↑ (¬ d)) ↑ ((¬ a) ↑b) ↑ ((¬ a) ↑d) ↑ ((¬ b) ↑e) ↑ ((¬ c) ↑e) ↑ (d↑e)
((¬ a) ↓ (¬ b) ↓ (¬ c) ↓d) ↓ ((¬ a) ↓ (¬ b) ↓e) ↓ ((¬ a) ↓ (¬ d) ↓e) ↓ (a↓b↓d↓e)
(e∧¬ c) ⊕ (c∧d∧e) ⊕ (d∧¬ a∧¬ e) ⊕ (a∧¬ b∧¬ d∧¬ e) ⊕ (b∧c∧¬ a∧¬ d) ⊕ (c∧e∧¬ b∧¬ d) ⊕ (b∧¬ a∧¬ c∧¬ d∧¬ e)

Resolvents, Resolution Sequences and ResolutionDepth

Resolvents

Resolvents[x, options] generates from the list x of clauses the list xx of all resolvents with respect to the list of logical variables given by one of the optional arguments, from which, however, some clauses are

excluded: with the option Reduce ->False, there are excluded clauses that are members of x, and with the option Reduce ->True there are excluded clauses a part of which is a member of x or xx.
Alternatives for options (default value = the first alternative) :
- Reduce -> True | False,
- Sort -> DeleteDuplicates | Union,
- Variables -> All | list of logical variables.

```
M = {q ∨ r ∨ s, r ∨ ¬ t, ¬ s ∨ t, q ∨ ¬ r ∨ s, q ∨ t, r ∨ ¬ s}
```

```
{q ∨ r ∨ s, r ∨ ¬ t, ¬ s ∨ t, q ∨ ¬ r ∨ s, q ∨ t, r ∨ ¬ s}
```

```
ClauseQ /@ M
```

```
{True, True, True, True, True, True}
```

```
Resolvents[M, Reduce → False, Sort → #] & /@ {DeleteDuplicates, Union} // TableForm
```

```
q ∨ s    q ∨ s ∨ ¬ t    q ∨ r ∨ t    q ∨ t ∨ ¬ r    q ∨ r
q ∨ r    q ∨ s          q ∨ r ∨ t    q ∨ s ∨ ¬ t    q ∨ t ∨ ¬ r
```

```
Resolvents[M, Reduce → #, Sort → Union] & /@ {False, True} // TableForm
```

```
q ∨ r    q ∨ s    q ∨ r ∨ t    q ∨ s ∨ ¬ t    q ∨ t ∨ ¬ r
q ∨ r    q ∨ s
```

```
Resolvents[M // Most, Reduce → #, Sort → Union] & /@ {False, True} // TableForm
```

```
q ∨ r    q ∨ s    r ∨ ¬ s    q ∨ r ∨ t    q ∨ s ∨ ¬ t    q ∨ t ∨ ¬ r
q ∨ r    q ∨ s    r ∨ ¬ s
```

```
Resolvents[M // Most // Most, Reduce → #, Sort → Union] & /@ {False, True} // TableForm
```

```
q ∨ s    r ∨ ¬ s    q ∨ r ∨ t    q ∨ s ∨ ¬ t    q ∨ t ∨ ¬ r
q ∨ s    r ∨ ¬ s    q ∨ r ∨ t    q ∨ t ∨ ¬ r
```

```
Resolvents[M // Most // Most, Reduce → #, Sort → Union, Variables → {q, s}] & /@ {False, True} //
  TableForm
```

```
q ∨ r ∨ t    q ∨ t ∨ ¬ r
q ∨ r ∨ t    q ∨ t ∨ ¬ r
```

FullResolutionSequence and ResolutionDepth

FullResolutionSequence[x, options], where x is a list of clauses, returns a finite sequence x, R[1,x], R[2,x], R[3,x],..., R[n,x], where R[k,x] is the list of all resolvents generated from clauses in the list Join[x,R[1,x],...,R[k-1,x]] and satisfying certain condition determined by options Reduce and Rules. The last member is either empty or contains FALSE. In the case Rules -> True the output contains also certain rules that are generated and applied before each step R[k,x] in order to reduce the number of clauses and to find truth values for logical variables in case the list x is satisfiable. In general case, however, the rules found may not be sufficient to transform all the formulae from x to True.
Alternatives for options (default value = the first alternative) :
- BooleanValues -> { 0, 1} | {False,True} | -> {FalseSymbol, TrueSymbol},
- Print -> False | True
- Reduce -> True | False,
- Rules -> True | False,
- Sort -> DeleteDuplicates | Union,
- Variables -> All | list of logical variables.

```
Options[FullResolutionSequence]
```

```
{BooleanValues → {0, 1}, Print → False, Reduce → True,
 Rules → True, Sort → DeleteDuplicates, Variables → All}
```

ResolutionDepth[x, options] characterizes the complexity of the list x of clauses by the triple depth={±n,r,t}, where n is the length of the list FullResolutionSequence[x,options], r is the total number

of clauses in it, and t is the CPU time spent in the *Mathematica* kernel. The sign is "+" if the list x is satisfiable, and "-" in the oppsite case.
Alternatives for options (default value = the first alternative) :
- Reduce -> True | False,
- Rules -> True | False,
- Sort -> DeleteDuplicates | Union,
- Trace -> True | False,
- Variables -> All | list of logical variables.

```
Options[ResolutionDepth]
```

```
{Reduce → True, Rules → True, Sort → False, Trace → True, Variables → All}
```

Example 1

```
M1 = {p ∨ q ∨ ¬ r, ¬ p ∨ s ∨ ¬ t, r ∨ t, ¬ s ∨ t, ¬ p ∨ ¬ u, ¬ q ∨ w, ¬ q ∨ ¬ w};
FullResolutionSequence[M1, Reduce → True, Rules → True, Sort → DeleteDuplicates]
```

{p ∨ q ∨ ¬ r, s ∨ ¬ p ∨ ¬ t, r ∨ t, t ∨ ¬ s, ¬ p ∨ ¬ u, w ∨ ¬ q, ¬ q ∨ ¬ w}
{u → 0}
{p ∨ q ∨ ¬ r, s ∨ ¬ p ∨ ¬ t, r ∨ t, t ∨ ¬ s, w ∨ ¬ q, ¬ q ∨ ¬ w}
{q ∨ s ∨ ¬ r ∨ ¬ t, p ∨ w ∨ ¬ r, p ∨ ¬ r ∨ ¬ w, p ∨ q ∨ t, r ∨ s ∨ ¬ p, ¬ q}
{q → 0, s → 1}
{p ∨ w ∨ ¬ r, p ∨ ¬ r ∨ ¬ w, p ∨ t}
{p → 1, r → 0, t → 1}
{}

```
ClearSystemCache[];
ResolutionDepth[M1, Reduce → True, Rules → True, Trace → #] & /@ {True, False} //
 Column[#, Center] &
```

```
{{0, 7, 0}, {1, 6, 0.}, {2, 6, 0.}, {3, 3, 0.}, {4, 0, 0.}}
{4, 22, 0.015625}
```

```
FullResolutionSequence[M1, Reduce → True, Rules → True, Sort → Union]
```

{r ∨ t, t ∨ ¬ s, w ∨ ¬ q, ¬ p ∨ ¬ u, ¬ q ∨ ¬ w, p ∨ q ∨ ¬ r, s ∨ ¬ p ∨ ¬ t}
{u → 0}
{r ∨ t, t ∨ ¬ s, w ∨ ¬ q, ¬ q ∨ ¬ w, p ∨ q ∨ ¬ r, s ∨ ¬ p ∨ ¬ t}
{¬ q, p ∨ q ∨ t, p ∨ w ∨ ¬ r, p ∨ ¬ r ∨ ¬ w, r ∨ s ∨ ¬ p, q ∨ s ∨ ¬ r ∨ ¬ t}
{q → 0, s → 1}
{p ∨ t, p ∨ w ∨ ¬ r, p ∨ ¬ r ∨ ¬ w}
{p → 1, r → 0, t → 1}
{}

```
ClearSystemCache[];
ResolutionDepth[M1, Reduce → True, Rules → True, Trace → #] & /@ {True, False} //
 Column[#, Center] &
```

```
{{0, 7, 0}, {1, 6, 0.}, {2, 6, 0.}, {3, 3, 0.}, {4, 0, 0.}}
{4, 22, 0.015625}
```

Example 2

```
M1 = {p ∨ q ∨ ¬ r, ¬ p ∨ s ∨ ¬ t, r ∨ t, ¬ s ∨ t, ¬ p ∨ ¬ u, ¬ q ∨ w, ¬ q ∨ ¬ w};
FullResolutionSequence[M1, Reduce → True, Rules → False, Sort → DeleteDuplicates]
```

{p ∨ q ∨ ¬ r, s ∨ ¬ p ∨ ¬ t, r ∨ t, t ∨ ¬ s, ¬ p ∨ ¬ u, w ∨ ¬ q, ¬ q ∨ ¬ w}
{q ∨ s ∨ ¬ r ∨ ¬ t, q ∨ ¬ r ∨ ¬ u, p ∨ w ∨ ¬ r, p ∨ ¬ r ∨ ¬ w, p ∨ q ∨ t, r ∨ s ∨ ¬ p, ¬ q}
{q ∨ t ∨ ¬ u, p ∨ ¬ r, s ∨ ¬ r ∨ ¬ t, ¬ r ∨ ¬ u, p ∨ t}
{t ∨ ¬ u}
{}

```
ClearSystemCache[];
ResolutionDepth[M1, Reduce → True, Rules → False, Sort → DeleteDuplicates, Trace → #] & /@
  {True, False} // Column[#, Center] &
```

{{0, 7, 0}, {1, 7, 0.}, {2, 5, 0.}, {3, 1, 0.015625}, {4, 0, 0.}}
{4, 20, 0.}

```
FullResolutionSequence[M1, Reduce → True, Rules → False, Sort → Union]
```

{r ∨ t, t ∨ ¬ s, w ∨ ¬ q, ¬ p ∨ ¬ u, ¬ q ∨ ¬ w, p ∨ q ∨ ¬ r, s ∨ ¬ p ∨ ¬ t}
{¬ q, p ∨ q ∨ t, p ∨ w ∨ ¬ r, p ∨ ¬ r ∨ ¬ w, q ∨ ¬ r ∨ ¬ u, r ∨ s ∨ ¬ p, q ∨ s ∨ ¬ r ∨ ¬ t}
{p ∨ t, p ∨ ¬ r, ¬ r ∨ ¬ u, q ∨ t ∨ ¬ u, s ∨ ¬ r ∨ ¬ t}
{t ∨ ¬ u}
{}

```
ClearSystemCache[];
ResolutionDepth[M1, Reduce → True, Rules → False, Trace → #] & /@ {True, False} //
 Column[#, Center] &
```

{{0, 7, 0}, {1, 7, 0.}, {2, 7, 0.}, {3, 6, 0.015625}, {4, 0, 0.}}
{4, 27, 0.015625}

Example 3

```
M1 = {p ∨ q ∨ ¬ r, ¬ p ∨ s ∨ ¬ t, r ∨ t, ¬ s ∨ t, ¬ p ∨ ¬ u, ¬ q ∨ w, ¬ q ∨ ¬ w};
FullResolutionSequence[M1, Reduce → False, Rules → True, Sort → DeleteDuplicates]
```

{p ∨ q ∨ ¬ r, s ∨ ¬ p ∨ ¬ t, r ∨ t, t ∨ ¬ s, ¬ p ∨ ¬ u, w ∨ ¬ q, ¬ q ∨ ¬ w}
{u → 0}
{p ∨ q ∨ ¬ r, s ∨ ¬ p ∨ ¬ t, r ∨ t, t ∨ ¬ s, w ∨ ¬ q, ¬ q ∨ ¬ w}
{q ∨ s ∨ ¬ r ∨ ¬ t, p ∨ w ∨ ¬ r, p ∨ ¬ r ∨ ¬ w, p ∨ q ∨ t, r ∨ s ∨ ¬ p, ¬ q}
{q → 0, s → 1}
{p ∨ w ∨ ¬ r, p ∨ ¬ r ∨ ¬ w, p ∨ t}
{p → 1, r → 0, t → 1}
{}

```
ClearSystemCache[];
ResolutionDepth[M1, Reduce → False, Rules → True, Sort → DeleteDuplicates, Trace → #] & /@
  {True, False} // Column[#, Center] &
```

{{0, 7, 0}, {1, 6, 0.}, {2, 6, 0.}, {3, 3, 0.}, {4, 0, 0.}}
{4, 22, 0.015625}

```
FullResolutionSequence[M1, Reduce → False, Rules → True, Sort → Union]
```

{r ∨ t, t ∨ ¬ s, w ∨ ¬ q, ¬ p ∨ ¬ u, ¬ q ∨ ¬ w, p ∨ q ∨ ¬ r, s ∨ ¬ p ∨ ¬ t}
{u → 0}
{r ∨ t, t ∨ ¬ s, w ∨ ¬ q, ¬ q ∨ ¬ w, p ∨ q ∨ ¬ r, s ∨ ¬ p ∨ ¬ t}
{¬ q, p ∨ q ∨ t, p ∨ w ∨ ¬ r, p ∨ ¬ r ∨ ¬ w, r ∨ s ∨ ¬ p, q ∨ s ∨ ¬ r ∨ ¬ t}
{q → 0, s → 1}
{p ∨ t, p ∨ w ∨ ¬ r, p ∨ ¬ r ∨ ¬ w}
{p → 1, r → 0, t → 1}
{}

```
ClearSystemCache[];
ResolutionDepth[M1, Reduce → False, Rules → True, Trace → #] & /@ {True, False} //
 Column[#, Center] &
```

```
{{0, 7, 0}, {1, 6, 0.}, {2, 6, 0.}, {3, 3, 0.}, {4, 0, 0.}}
                        {4, 22, 0.}
```

Example 4

```
M1 = {p ∨ q ∨ ¬ r, ¬ p ∨ s ∨ ¬ t, r ∨ t, ¬ s ∨ t, ¬ p ∨ ¬ u, ¬ q ∨ w, ¬ q ∨ ¬ w};
FullResolutionSequence[M1, Print → True, Reduce → False, Rules → False, Sort → Union]
```

```
{r ∨ t, t ∨ ¬ s, w ∨ ¬ q, ¬ p ∨ ¬ u, ¬ q ∨ ¬ w, p ∨ q ∨ ¬ r, s ∨ ¬ p ∨ ¬ t}
```

```
{¬ q, p ∨ q ∨ t, p ∨ w ∨ ¬ r, p ∨ ¬ r ∨ ¬ w, q ∨ ¬ r ∨ ¬ u, r ∨ s ∨ ¬ p, q ∨ s ∨ ¬ r ∨ ¬ t}
```

```
{p ∨ t, p ∨ ¬ r, ¬ r ∨ ¬ u, p ∨ t ∨ w, p ∨ t ∨ ¬ w, p ∨ ¬ q ∨ ¬ r, q ∨ t ∨ ¬ u, r ∨ t ∨ ¬ p, s ∨ ¬ r ∨ ¬ t, w ∨ ¬ r ∨ ¬ u,
 ¬ r ∨ ¬ u ∨ ¬ w, p ∨ q ∨ s ∨ ¬ r, q ∨ r ∨ s ∨ t, q ∨ s ∨ ¬ p ∨ ¬ t, q ∨ s ∨ ¬ p ∨ ¬ u, s ∨ w ∨ ¬ r ∨ ¬ t, s ∨ ¬ r ∨ ¬ t ∨ ¬ w}
```

```
{t ∨ ¬ u, p ∨ s ∨ ¬ r, p ∨ t ∨ ¬ q, p ∨ t ∨ ¬ r, p ∨ ¬ r ∨ ¬ u, q ∨ r ∨ t, r ∨ s ∨ t, r ∨ t ∨ w, r ∨ t ∨ ¬ w, s ∨ ¬ p ∨ ¬ u,
 t ∨ w ∨ ¬ u, t ∨ ¬ p ∨ ¬ u, t ∨ ¬ u ∨ ¬ w, ¬ q ∨ ¬ r ∨ ¬ u, p ∨ q ∨ s ∨ t, p ∨ q ∨ t ∨ ¬ r, p ∨ s ∨ w ∨ ¬ r,
 p ∨ s ∨ ¬ r ∨ ¬ t, p ∨ s ∨ ¬ r ∨ ¬ w, p ∨ t ∨ ¬ r ∨ ¬ u, q ∨ r ∨ s ∨ ¬ p, q ∨ s ∨ t ∨ ¬ u, q ∨ s ∨ ¬ r ∨ ¬ u,
 q ∨ t ∨ ¬ p ∨ ¬ u, r ∨ s ∨ t ∨ w, r ∨ s ∨ t ∨ ¬ w, s ∨ w ∨ ¬ p ∨ ¬ t, s ∨ w ∨ ¬ p ∨ ¬ u, s ∨ ¬ p ∨ ¬ t ∨ ¬ w,
 s ∨ ¬ p ∨ ¬ u ∨ ¬ w, s ∨ ¬ q ∨ ¬ r ∨ ¬ t, s ∨ ¬ r ∨ ¬ t ∨ ¬ u, t ∨ w ∨ ¬ p ∨ ¬ u, t ∨ ¬ p ∨ ¬ u ∨ ¬ w,
 p ∨ q ∨ s ∨ t ∨ w, p ∨ q ∨ s ∨ t ∨ ¬ w, p ∨ q ∨ s ∨ w ∨ ¬ r, p ∨ q ∨ s ∨ ¬ r ∨ ¬ w, q ∨ s ∨ t ∨ w ∨ ¬ u,
 q ∨ s ∨ t ∨ ¬ u ∨ ¬ w, q ∨ s ∨ w ∨ ¬ r ∨ ¬ t, q ∨ s ∨ w ∨ ¬ r ∨ ¬ u, q ∨ s ∨ ¬ r ∨ ¬ t ∨ ¬ w, q ∨ s ∨ ¬ r ∨ ¬ u ∨ ¬ w}
```

```
{p ∨ r ∨ t, p ∨ s ∨ t, p ∨ t ∨ ¬ u, r ∨ t ∨ ¬ q, r ∨ t ∨ ¬ u, s ∨ t ∨ ¬ u, s ∨ ¬ r ∨ ¬ u, t ∨ ¬ q ∨ ¬ u, t ∨ ¬ r ∨ ¬ u,
 p ∨ q ∨ t ∨ w, p ∨ q ∨ t ∨ ¬ u, p ∨ q ∨ t ∨ ¬ w, p ∨ r ∨ s ∨ t, p ∨ s ∨ t ∨ w, p ∨ s ∨ t ∨ ¬ q, p ∨ s ∨ t ∨ ¬ r,
 p ∨ s ∨ t ∨ ¬ u, p ∨ s ∨ t ∨ ¬ w, p ∨ s ∨ ¬ q ∨ ¬ r, p ∨ s ∨ ¬ r ∨ ¬ u, p ∨ t ∨ w ∨ ¬ q, p ∨ t ∨ w ∨ ¬ r, p ∨ t ∨ w ∨ ¬ u,
 p ∨ t ∨ ¬ q ∨ ¬ w, p ∨ t ∨ ¬ r ∨ ¬ w, p ∨ t ∨ ¬ u ∨ ¬ w, q ∨ r ∨ t ∨ ¬ p, q ∨ t ∨ w ∨ ¬ u, q ∨ t ∨ ¬ r ∨ ¬ u,
 q ∨ t ∨ ¬ u ∨ ¬ w, r ∨ s ∨ t ∨ ¬ q, r ∨ s ∨ t ∨ ¬ u, r ∨ s ∨ w ∨ ¬ p, r ∨ s ∨ ¬ p ∨ ¬ w, r ∨ t ∨ ¬ p ∨ ¬ u, s ∨ t ∨ w ∨ ¬ u,
 s ∨ t ∨ ¬ p ∨ ¬ u, s ∨ t ∨ ¬ q ∨ ¬ u, s ∨ t ∨ ¬ r ∨ ¬ u, s ∨ t ∨ ¬ u ∨ ¬ w, s ∨ w ∨ ¬ r ∨ ¬ u, s ∨ ¬ p ∨ ¬ q ∨ ¬ t,
 s ∨ ¬ p ∨ ¬ q ∨ ¬ u, s ∨ ¬ p ∨ ¬ r ∨ ¬ t, s ∨ ¬ p ∨ ¬ r ∨ ¬ u, s ∨ ¬ p ∨ ¬ t ∨ ¬ u, s ∨ ¬ q ∨ ¬ r ∨ ¬ u, s ∨ ¬ r ∨ ¬ u ∨ ¬ w,
 t ∨ w ∨ ¬ q ∨ ¬ u, t ∨ w ∨ ¬ r ∨ ¬ u, t ∨ ¬ p ∨ ¬ q ∨ ¬ u, t ∨ ¬ p ∨ ¬ r ∨ ¬ u, t ∨ ¬ q ∨ ¬ r ∨ ¬ u, t ∨ ¬ q ∨ ¬ u ∨ ¬ w,
 t ∨ ¬ r ∨ ¬ u ∨ ¬ w, p ∨ q ∨ r ∨ s ∨ t, p ∨ q ∨ s ∨ t ∨ ¬ r, p ∨ q ∨ s ∨ t ∨ ¬ u, p ∨ q ∨ s ∨ ¬ r ∨ ¬ t, p ∨ q ∨ s ∨ ¬ r ∨ ¬ u,
 p ∨ q ∨ t ∨ w ∨ ¬ r, p ∨ q ∨ t ∨ ¬ r ∨ ¬ w, p ∨ s ∨ t ∨ w ∨ ¬ q, p ∨ s ∨ t ∨ w ∨ ¬ r, p ∨ s ∨ t ∨ w ∨ ¬ u,
 p ∨ s ∨ t ∨ ¬ q ∨ ¬ w, p ∨ s ∨ t ∨ ¬ r ∨ ¬ u, p ∨ s ∨ t ∨ ¬ r ∨ ¬ w, p ∨ s ∨ t ∨ ¬ u ∨ ¬ w, p ∨ s ∨ w ∨ ¬ q ∨ ¬ r,
 p ∨ s ∨ w ∨ ¬ r ∨ ¬ t, p ∨ s ∨ w ∨ ¬ r ∨ ¬ u, p ∨ s ∨ ¬ q ∨ ¬ r ∨ ¬ u, p ∨ s ∨ ¬ q ∨ ¬ r ∨ ¬ w, p ∨ s ∨ ¬ r ∨ ¬ t ∨ ¬ w,
 p ∨ s ∨ ¬ r ∨ ¬ u ∨ ¬ w, q ∨ r ∨ s ∨ t ∨ w, q ∨ r ∨ s ∨ t ∨ ¬ u, q ∨ r ∨ s ∨ t ∨ ¬ w, q ∨ r ∨ s ∨ w ∨ ¬ p,
 q ∨ r ∨ s ∨ ¬ p ∨ ¬ w, q ∨ s ∨ t ∨ ¬ p ∨ ¬ u, q ∨ s ∨ t ∨ ¬ r ∨ ¬ u, q ∨ s ∨ w ∨ ¬ p ∨ ¬ t, q ∨ s ∨ w ∨ ¬ p ∨ ¬ u,
 q ∨ s ∨ ¬ p ∨ ¬ r ∨ ¬ t, q ∨ s ∨ ¬ p ∨ ¬ r ∨ ¬ u, q ∨ s ∨ ¬ p ∨ ¬ t ∨ ¬ u, q ∨ s ∨ ¬ p ∨ ¬ t ∨ ¬ w, q ∨ s ∨ ¬ p ∨ ¬ u ∨ ¬ w,
 q ∨ s ∨ ¬ r ∨ ¬ t ∨ ¬ u, q ∨ t ∨ w ∨ ¬ r ∨ ¬ u, q ∨ t ∨ ¬ r ∨ ¬ u ∨ ¬ w, r ∨ s ∨ t ∨ ¬ p ∨ ¬ u, s ∨ t ∨ w ∨ ¬ q ∨ ¬ u,
 s ∨ t ∨ w ∨ ¬ r ∨ ¬ u, s ∨ t ∨ ¬ q ∨ ¬ u ∨ ¬ w, s ∨ t ∨ ¬ r ∨ ¬ u ∨ ¬ w, s ∨ w ∨ ¬ p ∨ ¬ r ∨ ¬ u, s ∨ w ∨ ¬ q ∨ ¬ r ∨ ¬ t,
 s ∨ w ∨ ¬ q ∨ ¬ r ∨ ¬ u, s ∨ w ∨ ¬ r ∨ ¬ t ∨ ¬ u, s ∨ ¬ p ∨ ¬ q ∨ ¬ r ∨ ¬ u, s ∨ ¬ p ∨ ¬ r ∨ ¬ t ∨ ¬ u, s ∨ ¬ p ∨ ¬ r ∨ ¬ u ∨ ¬ w,
 s ∨ ¬ q ∨ ¬ r ∨ ¬ t ∨ ¬ w, s ∨ ¬ q ∨ ¬ r ∨ ¬ u ∨ ¬ w, s ∨ ¬ r ∨ ¬ t ∨ ¬ u ∨ ¬ w, t ∨ w ∨ ¬ q ∨ ¬ r ∨ ¬ u,
 t ∨ ¬ q ∨ ¬ r ∨ ¬ u ∨ ¬ w, p ∨ q ∨ s ∨ t ∨ ¬ r ∨ ¬ u, p ∨ q ∨ s ∨ w ∨ ¬ r ∨ ¬ u, p ∨ q ∨ s ∨ ¬ r ∨ ¬ u ∨ ¬ w,
 p ∨ s ∨ t ∨ w ∨ ¬ r ∨ ¬ u, p ∨ s ∨ t ∨ ¬ r ∨ ¬ u ∨ ¬ w, q ∨ s ∨ t ∨ w ∨ ¬ p ∨ ¬ u, q ∨ s ∨ t ∨ w ∨ ¬ r ∨ ¬ u,
 q ∨ s ∨ t ∨ ¬ p ∨ ¬ r ∨ ¬ u, q ∨ s ∨ t ∨ ¬ p ∨ ¬ u ∨ ¬ w, q ∨ s ∨ t ∨ ¬ r ∨ ¬ u ∨ ¬ w, q ∨ s ∨ w ∨ ¬ p ∨ ¬ r ∨ ¬ u,
 q ∨ s ∨ ¬ p ∨ ¬ r ∨ ¬ t ∨ ¬ u, q ∨ s ∨ ¬ p ∨ ¬ r ∨ ¬ u ∨ ¬ w, s ∨ w ∨ ¬ p ∨ ¬ q ∨ ¬ r ∨ ¬ u, s ∨ ¬ p ∨ ¬ q ∨ ¬ r ∨ ¬ u ∨ ¬ w}
```

```
{p ∨ q ∨ r ∨ t, p ∨ r ∨ t ∨ w, p ∨ r ∨ t ∨ ¬ q, p ∨ r ∨ t ∨ ¬ u, p ∨ r ∨ t ∨ ¬ w, p ∨ t ∨ ¬ q ∨ ¬ r, p ∨ t ∨ ¬ q ∨ ¬ u,
 q ∨ r ∨ t ∨ w, q ∨ r ∨ t ∨ ¬ u, q ∨ r ∨ t ∨ ¬ w, r ∨ s ∨ t ∨ ¬ p, r ∨ s ∨ ¬ p ∨ ¬ q, r ∨ s ∨ ¬ p ∨ ¬ t, r ∨ s ∨ ¬ p ∨ ¬ u,
 r ∨ t ∨ w ∨ ¬ p, r ∨ t ∨ w ∨ ¬ q, r ∨ t ∨ w ∨ ¬ u, r ∨ t ∨ ¬ p ∨ ¬ w, r ∨ t ∨ ¬ q ∨ ¬ u, r ∨ t ∨ ¬ q ∨ ¬ w,
 r ∨ t ∨ ¬ u ∨ ¬ w, p ∨ q ∨ t ∨ w ∨ ¬ u, p ∨ q ∨ t ∨ ¬ r ∨ ¬ u, p ∨ q ∨ t ∨ ¬ u ∨ ¬ w, p ∨ r ∨ s ∨ t ∨ w, p ∨ r ∨ s ∨ t ∨ ¬ q,
 p ∨ r ∨ s ∨ t ∨ ¬ u, p ∨ r ∨ s ∨ t ∨ ¬ w, p ∨ s ∨ t ∨ ¬ q ∨ ¬ r, p ∨ s ∨ t ∨ ¬ q ∨ ¬ u, p ∨ s ∨ ¬ q ∨ ¬ r ∨ ¬ t,
 p ∨ s ∨ ¬ r ∨ ¬ t ∨ ¬ u, p ∨ t ∨ w ∨ ¬ q ∨ ¬ r, p ∨ t ∨ w ∨ ¬ q ∨ ¬ u, p ∨ t ∨ w ∨ ¬ r ∨ ¬ u, p ∨ t ∨ ¬ q ∨ ¬ r ∨ ¬ u,
 p ∨ t ∨ ¬ q ∨ ¬ r ∨ ¬ w, p ∨ t ∨ ¬ q ∨ ¬ u ∨ ¬ w, p ∨ t ∨ ¬ r ∨ ¬ u ∨ ¬ w, q ∨ r ∨ s ∨ t ∨ ¬ p, q ∨ r ∨ s ∨ ¬ p ∨ ¬ t,
 q ∨ r ∨ s ∨ ¬ p ∨ ¬ u, q ∨ r ∨ t ∨ w ∨ ¬ p, q ∨ r ∨ t ∨ w ∨ ¬ u, q ∨ r ∨ t ∨ ¬ p ∨ ¬ w, q ∨ r ∨ t ∨ ¬ u ∨ ¬ w,
 q ∨ t ∨ w ∨ ¬ p ∨ ¬ u, q ∨ t ∨ ¬ p ∨ ¬ r ∨ ¬ u, q ∨ t ∨ ¬ p ∨ ¬ u ∨ ¬ w, r ∨ s ∨ t ∨ w ∨ ¬ p, r ∨ s ∨ t ∨ w ∨ ¬ q,
 r ∨ s ∨ t ∨ w ∨ ¬ u, r ∨ s ∨ t ∨ ¬ p ∨ ¬ w, r ∨ s ∨ t ∨ ¬ q ∨ ¬ u, r ∨ s ∨ t ∨ ¬ q ∨ ¬ w, r ∨ s ∨ t ∨ ¬ u ∨ ¬ w,
 r ∨ s ∨ w ∨ ¬ p ∨ ¬ q, r ∨ s ∨ w ∨ ¬ p ∨ ¬ t, r ∨ s ∨ w ∨ ¬ p ∨ ¬ u, r ∨ s ∨ ¬ p ∨ ¬ q ∨ ¬ u, r ∨ s ∨ ¬ p ∨ ¬ q ∨ ¬ w,
 r ∨ s ∨ ¬ p ∨ ¬ t ∨ ¬ w, r ∨ s ∨ ¬ p ∨ ¬ u ∨ ¬ w, r ∨ t ∨ w ∨ ¬ p ∨ ¬ u, r ∨ t ∨ w ∨ ¬ q ∨ ¬ u, r ∨ t ∨ ¬ p ∨ ¬ u ∨ ¬ w,
 r ∨ t ∨ ¬ q ∨ ¬ u ∨ ¬ w, s ∨ t ∨ w ∨ ¬ p ∨ ¬ u, s ∨ t ∨ ¬ p ∨ ¬ q ∨ ¬ u, s ∨ t ∨ ¬ p ∨ ¬ r ∨ ¬ u, s ∨ t ∨ ¬ p ∨ ¬ u ∨ ¬ w,
 s ∨ t ∨ ¬ q ∨ ¬ r ∨ ¬ u, s ∨ w ∨ ¬ p ∨ ¬ q ∨ ¬ t, s ∨ w ∨ ¬ p ∨ ¬ q ∨ ¬ u, s ∨ w ∨ ¬ p ∨ ¬ r ∨ ¬ t, s ∨ w ∨ ¬ p ∨ ¬ t ∨ ¬ u,
 s ∨ ¬ p ∨ ¬ q ∨ ¬ r ∨ ¬ t, s ∨ ¬ p ∨ ¬ q ∨ ¬ t ∨ ¬ w, s ∨ ¬ p ∨ ¬ q ∨ ¬ u ∨ ¬ w, s ∨ ¬ p ∨ ¬ r ∨ ¬ t ∨ ¬ w,
 s ∨ ¬ p ∨ ¬ t ∨ ¬ u ∨ ¬ w, s ∨ ¬ q ∨ ¬ r ∨ ¬ t ∨ ¬ u, t ∨ w ∨ ¬ p ∨ ¬ q ∨ ¬ u, t ∨ w ∨ ¬ p ∨ ¬ r ∨ ¬ u,
 t ∨ ¬ p ∨ ¬ q ∨ ¬ r ∨ ¬ u, t ∨ ¬ p ∨ ¬ q ∨ ¬ u ∨ ¬ w, t ∨ ¬ p ∨ ¬ r ∨ ¬ u ∨ ¬ w, p ∨ q ∨ r ∨ s ∨ t ∨ ¬ u, p ∨ q ∨ s ∨ t ∨ w ∨ ¬ u,
 p ∨ q ∨ s ∨ t ∨ ¬ u ∨ ¬ w, p ∨ q ∨ s ∨ ¬ r ∨ ¬ t ∨ ¬ u, p ∨ q ∨ t ∨ w ∨ ¬ r ∨ ¬ u, p ∨ q ∨ t ∨ ¬ r ∨ ¬ u ∨ ¬ w,
 p ∨ r ∨ s ∨ t ∨ w ∨ ¬ u, p ∨ r ∨ s ∨ t ∨ ¬ u ∨ ¬ w, p ∨ s ∨ t ∨ w ∨ ¬ q ∨ ¬ u, p ∨ s ∨ t ∨ ¬ q ∨ ¬ r ∨ ¬ u,
 p ∨ s ∨ t ∨ ¬ q ∨ ¬ u ∨ ¬ w, p ∨ s ∨ w ∨ ¬ q ∨ ¬ r ∨ ¬ u, p ∨ s ∨ w ∨ ¬ r ∨ ¬ t ∨ ¬ u, p ∨ s ∨ ¬ q ∨ ¬ r ∨ ¬ t ∨ ¬ u,
 p ∨ s ∨ ¬ q ∨ ¬ r ∨ ¬ u ∨ ¬ w, p ∨ s ∨ ¬ r ∨ ¬ t ∨ ¬ u ∨ ¬ w, q ∨ r ∨ s ∨ t ∨ w ∨ ¬ u, q ∨ r ∨ s ∨ t ∨ ¬ p ∨ ¬ u,
 q ∨ r ∨ s ∨ t ∨ ¬ u ∨ ¬ w, q ∨ r ∨ s ∨ w ∨ ¬ p ∨ ¬ u, q ∨ r ∨ s ∨ ¬ p ∨ ¬ u ∨ ¬ w, q ∨ s ∨ w ∨ ¬ p ∨ ¬ t ∨ ¬ u,
 q ∨ s ∨ w ∨ ¬ r ∨ ¬ t ∨ ¬ u, q ∨ s ∨ ¬ p ∨ ¬ t ∨ ¬ u ∨ ¬ w, q ∨ s ∨ ¬ r ∨ ¬ t ∨ ¬ u ∨ ¬ w, q ∨ t ∨ w ∨ ¬ p ∨ ¬ r ∨ ¬ u,
 q ∨ t ∨ ¬ p ∨ ¬ r ∨ ¬ u ∨ ¬ w, r ∨ s ∨ t ∨ w ∨ ¬ p ∨ ¬ u, r ∨ s ∨ t ∨ w ∨ ¬ q ∨ ¬ u, r ∨ s ∨ t ∨ ¬ p ∨ ¬ q ∨ ¬ u,
 r ∨ s ∨ t ∨ ¬ p ∨ ¬ u ∨ ¬ w, r ∨ s ∨ t ∨ ¬ q ∨ ¬ u ∨ ¬ w, s ∨ t ∨ w ∨ ¬ p ∨ ¬ q ∨ ¬ u, s ∨ t ∨ w ∨ ¬ p ∨ ¬ r ∨ ¬ u,
 s ∨ t ∨ w ∨ ¬ q ∨ ¬ r ∨ ¬ u, s ∨ t ∨ ¬ p ∨ ¬ q ∨ ¬ r ∨ ¬ u, s ∨ t ∨ ¬ p ∨ ¬ q ∨ ¬ u ∨ ¬ w, s ∨ t ∨ ¬ p ∨ ¬ r ∨ ¬ u ∨ ¬ w,
 s ∨ t ∨ ¬ q ∨ ¬ r ∨ ¬ u ∨ ¬ w, s ∨ w ∨ ¬ p ∨ ¬ r ∨ ¬ t ∨ ¬ u, s ∨ w ∨ ¬ q ∨ ¬ r ∨ ¬ t ∨ ¬ u, s ∨ ¬ p ∨ ¬ q ∨ ¬ r ∨ ¬ t ∨ ¬ u,
 s ∨ ¬ p ∨ ¬ r ∨ ¬ t ∨ ¬ u ∨ ¬ w, s ∨ ¬ q ∨ ¬ r ∨ ¬ t ∨ ¬ u ∨ ¬ w, t ∨ w ∨ ¬ p ∨ ¬ q ∨ ¬ r ∨ ¬ u, t ∨ ¬ p ∨ ¬ q ∨ ¬ r ∨ ¬ u ∨ ¬ w}
```

```
{r ∨ t ∨ ¬ p ∨ ¬ q, p ∨ q ∨ r ∨ t ∨ ¬ u, p ∨ r ∨ t ∨ w ∨ ¬ u, p ∨ r ∨ t ∨ ¬ q ∨ ¬ u, p ∨ r ∨ t ∨ ¬ u ∨ ¬ w,
 q ∨ r ∨ t ∨ ¬ p ∨ ¬ u, r ∨ s ∨ t ∨ ¬ p ∨ ¬ q, r ∨ s ∨ ¬ p ∨ ¬ q ∨ ¬ t, r ∨ s ∨ ¬ p ∨ ¬ t ∨ ¬ u, r ∨ t ∨ w ∨ ¬ p ∨ ¬ q,
 r ∨ t ∨ ¬ p ∨ ¬ q ∨ ¬ u, r ∨ t ∨ ¬ p ∨ ¬ q ∨ ¬ w, s ∨ ¬ p ∨ ¬ q ∨ ¬ t ∨ ¬ u, p ∨ r ∨ s ∨ t ∨ ¬ q ∨ ¬ u,
 p ∨ t ∨ w ∨ ¬ q ∨ ¬ r ∨ ¬ u, p ∨ t ∨ ¬ q ∨ ¬ r ∨ ¬ u ∨ ¬ w, q ∨ r ∨ s ∨ ¬ p ∨ ¬ t ∨ ¬ u, q ∨ r ∨ t ∨ w ∨ ¬ p ∨ ¬ u,
 q ∨ r ∨ t ∨ ¬ p ∨ ¬ u ∨ ¬ w, r ∨ s ∨ w ∨ ¬ p ∨ ¬ q ∨ ¬ u, r ∨ s ∨ w ∨ ¬ p ∨ ¬ t ∨ ¬ u, r ∨ s ∨ ¬ p ∨ ¬ q ∨ ¬ t ∨ ¬ u,
 r ∨ s ∨ ¬ p ∨ ¬ q ∨ ¬ u ∨ ¬ w, r ∨ s ∨ ¬ p ∨ ¬ t ∨ ¬ u ∨ ¬ w, s ∨ w ∨ ¬ p ∨ ¬ q ∨ ¬ t ∨ ¬ u, s ∨ ¬ p ∨ ¬ q ∨ ¬ t ∨ ¬ u ∨ ¬ w}
```

```
{r ∨ t ∨ w ∨ ¬ p ∨ ¬ q ∨ ¬ u, r ∨ t ∨ ¬ p ∨ ¬ q ∨ ¬ u ∨ ¬ w}

{}
```

```
ClearSystemCache[];
M1 = {p ∨ q ∨ ¬ r, ¬ p ∨ s ∨ ¬ t, r ∨ t, ¬ s ∨ t, ¬ p ∨ ¬ u, ¬ q ∨ w, ¬ q ∨ ¬ w};
ResolutionDepth[M1, Reduce → False, Rules → False, Sort → Union, Trace → #] & /@ {True, False} //
 Column[#, Center] &
```

```
{{0, 7, 0}, {1, 7, 0.}, {2, 17, 0.}, {3, 44, 0.03125}, {4, 125, 0.4375},
  {5, 133, 13.5}, {6, 26, 69.4375}, {7, 2, 25.4688}, {8, 0, 3.125}}
{8, 361, 114.516}
```

Example 5

```
M2 = {p ∨ s, p ∨ r ∨ t, q ∨ r, q ∨ s, q ∨ ¬ p ∨ t, r ∨ ¬ q ∨ t, s ∨ ¬ r, t ∨ ¬ s, ¬ t};
FullResolutionSequence[M2, Reduce → True, Rules → True, Sort → DeleteDuplicates]
```

{p ∨ s, p ∨ r ∨ t, q ∨ r, q ∨ s, q ∨ t ∨ ¬ p, r ∨ t ∨ ¬ q, s ∨ ¬ r, t ∨ ¬ s, ¬ t}
{t → 0}
{p ∨ s, p ∨ r, q ∨ r, q ∨ s, q ∨ ¬ p, r ∨ ¬ q, s ∨ ¬ r, ¬ s}
{s → 0}
{p, p ∨ r, q ∨ r, q, q ∨ ¬ p, r ∨ ¬ q, ¬ r}
{p → 1, q → 1, r → 0}
{FALSE}

```
FullResolutionSequence[M2, Reduce → True, Rules → True, Sort → Union]
```

{¬ t, p ∨ s, q ∨ r, q ∨ s, s ∨ ¬ r, t ∨ ¬ s, p ∨ r ∨ t, q ∨ t ∨ ¬ p, r ∨ t ∨ ¬ q}
{t → 0}
{¬ s, p ∨ r, p ∨ s, q ∨ r, q ∨ s, q ∨ ¬ p, r ∨ ¬ q, s ∨ ¬ r}
{s → 0}
{p, q, ¬ r, p ∨ r, q ∨ r, q ∨ ¬ p, r ∨ ¬ q}
{p → 1, q → 1, r → 0}
{FALSE}

```
ClearSystemCache[];
ResolutionDepth[M2, Reduce → True, Rules → True, Trace → #] & /@ {True, False} //
 Column[#, Center] &
```

```
{{0, 9, 0}, {1, 8, 0.}, {2, 7, 0.}, {-3, 1, 0.}}
{-3, 25, 0.}
```

Example 6

```
M2 = {p ∨ s, p ∨ r ∨ t, q ∨ r, q ∨ s, q ∨ ¬ p ∨ t, r ∨ ¬ q ∨ t, s ∨ ¬ r, t ∨ ¬ s, ¬ t};
FullResolutionSequence[M2, Reduce → True, Rules → False, Sort → DeleteDuplicates]
```

{p ∨ s, p ∨ r ∨ t, q ∨ r, q ∨ s, q ∨ t ∨ ¬ p, r ∨ t ∨ ¬ q, s ∨ ¬ r, t ∨ ¬ s, ¬ t}
{r ∨ t, s ∨ t ∨ ¬ q, p ∨ t, q ∨ t, t ∨ ¬ r, p ∨ r, q ∨ ¬ p, r ∨ ¬ q, ¬ s}
{r, s ∨ ¬ q, t, p, q, ¬ r}
{s, ¬ q, FALSE}

```
FullResolutionSequence[M2, Reduce → True, Rules → False, Sort → Union]
```

{¬ t, p ∨ s, q ∨ r, q ∨ s, s ∨ ¬ r, t ∨ ¬ s, p ∨ r ∨ t, q ∨ t ∨ ¬ p, r ∨ t ∨ ¬ q}
{¬ s, p ∨ r, p ∨ t, q ∨ t, q ∨ ¬ p, r ∨ t, r ∨ ¬ q, t ∨ ¬ r, s ∨ t ∨ ¬ q}
{p, q, r, t, ¬ r, s ∨ ¬ q}
{FALSE, s, ¬ q}

```
ClearSystemCache[];
ResolutionDepth[M2, Reduce → True, Rules → False, Trace → #] & /@ {True, False} //
  Column[#, Center] &
```

{{0, 9, 0}, {1, 9, 0.}, {2, 11, 0.}, {-3, 22, 0.015625}}
{-3, 51, 0.}

Example 7

```
M2 = {p ∨ s, p ∨ r ∨ t, q ∨ r, q ∨ s, q ∨ ¬ p ∨ t, r ∨ ¬ q ∨ t, s ∨ ¬ r, t ∨ ¬ s, ¬ t};
FullResolutionSequence[M2, Reduce → False, Rules → True, Sort → DeleteDuplicates]
```

{p ∨ s, p ∨ r ∨ t, q ∨ r, q ∨ s, q ∨ t ∨ ¬ p, r ∨ t ∨ ¬ q, s ∨ ¬ r, t ∨ ¬ s, ¬ t}
{t → 0}
{p ∨ s, p ∨ r, q ∨ r, q ∨ s, q ∨ ¬ p, r ∨ ¬ q, s ∨ ¬ r, ¬ s}
{s → 0}
{p, p ∨ r, q ∨ r, q, q ∨ ¬ p, r ∨ ¬ q, ¬ r}
{p → 1, q → 1, r → 0}
{FALSE}

```
M2 = {p ∨ s, p ∨ r ∨ t, q ∨ r, q ∨ s, q ∨ ¬ p ∨ t, r ∨ ¬ q ∨ t, s ∨ ¬ r, t ∨ ¬ s, ¬ t};
FullResolutionSequence[M2, Reduce → False, Rules → True, Sort → Union]
```

{¬ t, p ∨ s, q ∨ r, q ∨ s, s ∨ ¬ r, t ∨ ¬ s, p ∨ r ∨ t, q ∨ t ∨ ¬ p, r ∨ t ∨ ¬ q}
{t → 0}
{¬ s, p ∨ r, p ∨ s, q ∨ r, q ∨ s, q ∨ ¬ p, r ∨ ¬ q, s ∨ ¬ r}
{s → 0}
{p, q, ¬ r, p ∨ r, q ∨ r, q ∨ ¬ p, r ∨ ¬ q}
{p → 1, q → 1, r → 0}
{FALSE}

```
ClearSystemCache[];
ResolutionDepth[M2, Reduce → False, Rules → True, Trace → #] & /@ {True, False} //
  Column[#, Center] &
```

{{0, 9, 0}, {1, 8, 0.}, {2, 7, 0.}, {-3, 1, 0.}}
{-3, 25, 0.}

Example 8

```
M2 = {p∨s, p∨r∨t, q∨r, q∨s, q∨¬p∨t, r∨¬q∨t, s∨¬r, t∨¬s, ¬t};
FullResolutionSequence[M2, Reduce → False, Rules → False, Sort → DeleteDuplicates]
```

{p∨s, p∨r∨t, q∨r, q∨s, q∨t∨¬p, r∨t∨¬q, s∨¬r, t∨¬s, ¬t}
{q∨s∨t, q∨r∨t, r∨t, r∨s∨t, r∨t∨¬p, p∨s∨t, s∨t∨¬q, p∨t, q∨t, t∨¬r, p∨r, q∨¬p, r∨¬q, ¬s}
{s∨t, s∨t∨¬p, r, r∨s, r∨¬p, t∨¬q, s∨¬q, t, t∨¬p, p, q, ¬r}
{s, s∨¬p, ¬q, FALSE, ¬p}

```
FullResolutionSequence[M2, Reduce → False, Rules → False, Sort → Union]
```

{¬t, p∨s, q∨r, q∨s, s∨¬r, t∨¬s, p∨r∨t, q∨t∨¬p, r∨t∨¬q}
{¬s, p∨r, p∨t, q∨t, q∨¬p, r∨t, r∨¬q, t∨¬r, p∨s∨t, q∨r∨t, q∨s∨t, r∨s∨t, r∨t∨¬p, s∨t∨¬q}
{p, q, r, t, ¬r, r∨s, r∨¬p, s∨t, s∨¬q, t∨¬p, t∨¬q, s∨t∨¬p}
{FALSE, s, ¬p, ¬q, s∨¬p}

```
ClearSystemCache[];
ResolutionDepth[M2, Reduce → False, Rules → False, Trace → #] & /@ {True, False} //
 Column[#, Center] &
```

```
{{0, 9, 0}, {1, 14, 0.}, {2, 29, 0.}, {-3, 68, 0.015625}}
{-3, 120, 0.015625}
```

ResolutionSequence

ResolutionSequence[x_List, options] tests whether the list x of clauses is satisfiable or not and in the positive case with the option Rules -> True finds, in some cases, values of logical variables for which all clauses in x are true. The algorithm used can be roughly described as a successive elimination of logical variables.
Alternatives for options (default value = the first alternative) :
- BooleanValues | {False -> 0, True -> 1} | {False -> FalseSymbol, True -> TrueSymbol},
- ItemSize -> Automatic | {width,height},
- Print -> False | True,
- Reduce -> True | False,
- Rules -> True | False,
- SelectionRule -> First | Last | Random,
- Sort -> DeleteDuplicates | Union,
- Spacings -> 1,
- SequenceBreaks -> None | integer > 1 | increasing list of integers > 1,
- Variables -> All | list of logical variables.

Example 1

```
M1 = {p∨q∨¬r, ¬p∨s∨¬t, r∨t, ¬s∨t, ¬p∨¬u, ¬q∨w, ¬q∨¬w};
ResolutionSequence[M1, ItemSize → 40, Print → True, Reduce → True, Rules → True]
```

{p ∨ q ∨ ¬ r, s ∨ ¬ p ∨ ¬ t, r ∨ t, t ∨ ¬ s, ¬ p ∨ ¬ u, w ∨ ¬ q, ¬ q ∨ ¬ w}
•p•
{{p ∨ q ∨ ¬ r}, {s ∨ ¬ p ∨ ¬ t, ¬ p ∨ ¬ u}} → {q ∨ s ∨ ¬ r ∨ ¬ t, q ∨ ¬ r ∨ ¬ u} , {r ∨ t, t ∨ ¬ s, w ∨ ¬ q, ¬ q ∨ ¬ w}
{q ∨ s ∨ ¬ r ∨ ¬ t, q ∨ ¬ r ∨ ¬ u, r ∨ t, t ∨ ¬ s, w ∨ ¬ q, ¬ q ∨ ¬ w}
•q•
{{q ∨ s ∨ ¬ r ∨ ¬ t, q ∨ ¬ r ∨ ¬ u}, {w ∨ ¬ q, ¬ q ∨ ¬ w}} → {s ∨ w ∨ ¬ r ∨ ¬ t, w ∨ ¬ r ∨ ¬ u, s ∨ ¬ r ∨ ¬ t ∨ ¬ w, ¬ r ∨ ¬ u ∨ ¬ w} , {r ∨ t, t ∨ ¬ s}
{s ∨ w ∨ ¬ r ∨ ¬ t, w ∨ ¬ r ∨ ¬ u, s ∨ ¬ r ∨ ¬ t ∨ ¬ w, ¬ r ∨ ¬ u ∨ ¬ w, r ∨ t, t ∨ ¬ s}
•r•
{{r ∨ t}, {s ∨ w ∨ ¬ r ∨ ¬ t, w ∨ ¬ r ∨ ¬ u, s ∨ ¬ r ∨ ¬ t ∨ ¬ w, ¬ r ∨ ¬ u ∨ ¬ w}} → {t ∨ w ∨ ¬ u, t ∨ ¬ u ∨ ¬ w} , {t ∨ ¬ s}
{t ∨ w ∨ ¬ u, t ∨ ¬ u ∨ ¬ w, t ∨ ¬ s}
•s → 0•
{t ∨ w ∨ ¬ u, t ∨ ¬ u ∨ ¬ w}
•t → 1•
{}
••{s → 0, t → 1}••
{p ∨ q ∨ ¬ r, ¬ p, ¬ p ∨ ¬ u, w ∨ ¬ q, ¬ q ∨ ¬ w}
•w•
{{w ∨ ¬ q}, {¬ q ∨ ¬ w}} → {¬ q} , {p ∨ q ∨ ¬ r, ¬ p, ¬ p ∨ ¬ u}
{¬ q, p ∨ q ∨ ¬ r, ¬ p, ¬ p ∨ ¬ u}
•u → 0•
{¬ q, p ∨ q ∨ ¬ r, ¬ p}
•r → 0•
{¬ q, ¬ p}
•q → 0•
{¬ p}
•p → 0•
{}
••{s → 0, t → 1, u → 0, r → 0, q → 0, p → 0}••

```
M1 = {p ∨ q ∨ ¬ r, ¬ p ∨ s ∨ ¬ t, r ∨ t, ¬ s ∨ t, ¬ p ∨ ¬ u, ¬ q ∨ w, ¬ q ∨ ¬ w};
ResolutionSequence[M1, ItemSize → {40, Automatic},
 Print → True, Reduce → False, Rules → True, SequenceBreaks → 7]
```

{p ∨ q ∨ ¬ r, s ∨ ¬ p ∨ ¬ t, r ∨ t, t ∨ ¬ s, ¬ p ∨ ¬ u, w ∨ ¬ q, ¬ q ∨ ¬ w}
•p•
{{p ∨ q ∨ ¬ r}, {s ∨ ¬ p ∨ ¬ t, ¬ p ∨ ¬ u}} → {q ∨ s ∨ ¬ r ∨ ¬ t, q ∨ ¬ r ∨ ¬ u} , {r ∨ t, t ∨ ¬ s, w ∨ ¬ q, ¬ q ∨ ¬ w}
{q ∨ s ∨ ¬ r ∨ ¬ t, q ∨ ¬ r ∨ ¬ u, r ∨ t, t ∨ ¬ s, w ∨ ¬ q, ¬ q ∨ ¬ w}
•q•
{{q ∨ s ∨ ¬ r ∨ ¬ t, q ∨ ¬ r ∨ ¬ u}, {w ∨ ¬ q, ¬ q ∨ ¬ w}} → {s ∨ w ∨ ¬ r ∨ ¬ t, w ∨ ¬ r ∨ ¬ u, s ∨ ¬ r ∨ ¬ t ∨ ¬ w, ¬ r ∨ ¬ u ∨ ¬ w} , {r ∨ t, t ∨ ¬ s}
{s ∨ w ∨ ¬ r ∨ ¬ t, w ∨ ¬ r ∨ ¬ u, s ∨ ¬ r ∨ ¬ t ∨ ¬ w, ¬ r ∨ ¬ u ∨ ¬ w, r ∨ t, t ∨ ¬ s}

•r•
{{r ∨ t}, {s ∨ w ∨ ¬ r ∨ ¬ t, w ∨ ¬ r ∨ ¬ u, s ∨ ¬ r ∨ ¬ t ∨ ¬ w, ¬ r ∨ ¬ u ∨ ¬ w}} → {t ∨ w ∨ ¬ u, t ∨ ¬ u ∨ ¬ w} , {t ∨ ¬ s}
{t ∨ w ∨ ¬ u, t ∨ ¬ u ∨ ¬ w, t ∨ ¬ s}
•s → 0•
{t ∨ w ∨ ¬ u, t ∨ ¬ u ∨ ¬ w}
•t → 1•
{}

••{s → 0, t → 1}••
{p ∨ q ∨ ¬ r, ¬ p, ¬ p ∨ ¬ u, w ∨ ¬ q, ¬ q ∨ ¬ w}
•w•
{{w ∨ ¬ q}, {¬ q ∨ ¬ w}} → {¬ q} , {p ∨ q ∨ ¬ r, ¬ p, ¬ p ∨ ¬ u}
{¬ q, p ∨ q ∨ ¬ r, ¬ p, ¬ p ∨ ¬ u}
•u → 0•
{¬ q, p ∨ q ∨ ¬ r, ¬ p}

•r → 0•
{¬ q, ¬ p}
•q → 0•
{¬ p}
•p → 0•
{}
••{s → 0, t → 1, u → 0, r → 0, q → 0, p → 0}••

```
M1 = {p ∨ q ∨ ¬ r, ¬ p ∨ s ∨ ¬ t, r ∨ t, ¬ s ∨ t, ¬ p ∨ ¬ u, ¬ q ∨ w, ¬ q ∨ ¬ w};
ResolutionSequence[M1, Reduce → True, Rules → False]
```

{p∨q∨¬r, s∨¬p∨¬t, r∨t, t∨¬s, ¬p∨¬u, w∨¬q, ¬q∨¬w}
•p•
{{p∨q∨¬r}, {s∨¬p∨¬t, ¬p∨¬u}} → {q∨s∨¬r∨¬t, q∨¬r∨¬u} , {r∨t, t∨¬s, w∨¬q, ¬q∨¬w}
{q∨s∨¬r∨¬t, q∨¬r∨¬u, r∨t, t∨¬s, w∨¬q, ¬q∨¬w}
•q•
{{q∨s∨¬r∨¬t, q∨¬r∨¬u}, {w∨¬q, ¬q∨¬w}} → {s∨w∨¬r∨¬t, w∨¬r∨¬u, s∨¬r∨¬t∨¬w, ¬r∨¬u∨¬w} , {r∨t, t∨¬s}
{s∨w∨¬r∨¬t, w∨¬r∨¬u, s∨¬r∨¬t∨¬w, ¬r∨¬u∨¬w, r∨t, t∨¬s}
•r•
{{r∨t}, {s∨w∨¬r∨¬t, w∨¬r∨¬u, s∨¬r∨¬t∨¬w, ¬r∨¬u∨¬w}} → {t∨w∨¬u, t∨¬u∨¬w} , {t∨¬s}
{t∨w∨¬u, t∨¬u∨¬w, t∨¬s}
•s•
{{t∨¬s}} , {t∨w∨¬u, t∨¬u∨¬w}
{t∨w∨¬u, t∨¬u∨¬w}
•w•
{{t∨w∨¬u}, {t∨¬u∨¬w}} → {t∨¬u}
{t∨¬u}

```
M1 = {p∨q∨¬r, ¬p∨s∨¬t, r∨t, ¬s∨t, ¬p∨¬u, ¬q∨w, ¬q∨¬w};
ResolutionSequence[M1, Reduce → False, Rules → False]
```

{p∨q∨¬r, s∨¬p∨¬t, r∨t, t∨¬s, ¬p∨¬u, w∨¬q, ¬q∨¬w}
•p•
{{p∨q∨¬r}, {s∨¬p∨¬t, ¬p∨¬u}} → {q∨s∨¬r∨¬t, q∨¬r∨¬u} , {r∨t, t∨¬s, w∨¬q, ¬q∨¬w}
{q∨s∨¬r∨¬t, q∨¬r∨¬u, r∨t, t∨¬s, w∨¬q, ¬q∨¬w}
•q•
{{q∨s∨¬r∨¬t, q∨¬r∨¬u}, {w∨¬q, ¬q∨¬w}} → {s∨w∨¬r∨¬t, w∨¬r∨¬u, s∨¬r∨¬t∨¬w, ¬r∨¬u∨¬w} , {r∨t, t∨¬s}
{s∨w∨¬r∨¬t, w∨¬r∨¬u, s∨¬r∨¬t∨¬w, ¬r∨¬u∨¬w, r∨t, t∨¬s}
•r•
{{r∨t}, {s∨w∨¬r∨¬t, w∨¬r∨¬u, s∨¬r∨¬t∨¬w, ¬r∨¬u∨¬w}} → {t∨w∨¬u, t∨¬u∨¬w} , {t∨¬s}
{t∨w∨¬u, t∨¬u∨¬w, t∨¬s}
•s•
{{t∨¬s}} , {t∨w∨¬u, t∨¬u∨¬w}
{t∨w∨¬u, t∨¬u∨¬w}
•w•
{{t∨w∨¬u}, {t∨¬u∨¬w}} → {t∨¬u}
{t∨¬u}

Example 2

```
M1 = {p ∨ q ∨ ¬ r, ¬ p ∨ s ∨ ¬ t, r ∨ t, ¬ s ∨ t, ¬ p ∨ ¬ u, ¬ q ∨ w, ¬ q ∨ ¬ w};
ResolutionSequence[M1, Reduce → True, Rules → True, SelectionRule → Last, Sort → Union]
```

{r ∨ t, t ∨ ¬ s, w ∨ ¬ q, ¬ p ∨ ¬ u, ¬ q ∨ ¬ w, p ∨ q ∨ ¬ r, s ∨ ¬ p ∨ ¬ t}
•w•
{{w ∨ ¬ q}, {¬ q ∨ ¬ w}} → {¬ q} , {r ∨ t, t ∨ ¬ s, ¬ p ∨ ¬ u, p ∨ q ∨ ¬ r, s ∨ ¬ p ∨ ¬ t}
{¬ q, r ∨ t, t ∨ ¬ s, ¬ p ∨ ¬ u, p ∨ q ∨ ¬ r, s ∨ ¬ p ∨ ¬ t}
•u → 0•
{¬ q, r ∨ t, t ∨ ¬ s, p ∨ q ∨ ¬ r, s ∨ ¬ p ∨ ¬ t}
•t•
{{r ∨ t, t ∨ ¬ s}, {s ∨ ¬ p ∨ ¬ t}} → {r ∨ s ∨ ¬ p} , {¬ q, p ∨ q ∨ ¬ r}
{¬ q, p ∨ q ∨ ¬ r, r ∨ s ∨ ¬ p}
•s → 1•
{¬ q, p ∨ q ∨ ¬ r}
•r → 0•
{¬ q}
•q → 0•
{}
••{q → 0, r → 0, s → 1, u → 0}••
{t}
•t → 1•
{}
••{q → 0, r → 0, s → 1, t → 1, u → 0}••

```
M1 = {p ∨ q ∨ ¬ r, ¬ p ∨ s ∨ ¬ t, r ∨ t, ¬ s ∨ t, ¬ p ∨ ¬ u, ¬ q ∨ w, ¬ q ∨ ¬ w};
ResolutionSequence[M1, ItemSize → {38, Automatic}, Print → True, Reduce → True,
 Rules → True, SelectionRule → Random, SequenceBreaks → 7, Sort → Union]
```

{r ∨ t, t ∨ ¬ s, w ∨ ¬ q, ¬ p ∨ ¬ u, ¬ q ∨ ¬ w, p ∨ q ∨ ¬ r, s ∨ ¬ p ∨ ¬ t}
•q•
{{p ∨ q ∨ ¬ r}, {w ∨ ¬ q, ¬ q ∨ ¬ w}} → {p ∨ w ∨ ¬ r, p ∨ ¬ r ∨ ¬ w} , {r ∨ t, t ∨ ¬ s, ¬ p ∨ ¬ u, s ∨ ¬ p ∨ ¬ t}
{r ∨ t, t ∨ ¬ s, ¬ p ∨ ¬ u, p ∨ w ∨ ¬ r, p ∨ ¬ r ∨ ¬ w, s ∨ ¬ p ∨ ¬ t}
•p•
{{p ∨ w ∨ ¬ r, p ∨ ¬ r ∨ ¬ w}, {¬ p ∨ ¬ u, s ∨ ¬ p ∨ ¬ t}} → {w ∨ ¬ r ∨ ¬ u, ¬ r ∨ ¬ u ∨ ¬ w, s ∨ w ∨ ¬ r ∨ ¬ t, s ∨ ¬ r ∨ ¬ t ∨ ¬ w} , {r ∨ t, t ∨ ¬ s}
{r ∨ t, t ∨ ¬ s, w ∨ ¬ r ∨ ¬ u, ¬ r ∨ ¬ u ∨ ¬ w, s ∨ w ∨ ¬ r ∨ ¬ t, s ∨ ¬ r ∨ ¬ t ∨ ¬ w}

•s•
{{s ∨ w ∨ ¬ r ∨ ¬ t, s ∨ ¬ r ∨ ¬ t ∨ ¬ w}, {t ∨ ¬ s}} → {} , {r ∨ t, w ∨ ¬ r ∨ ¬ u, ¬ r ∨ ¬ u ∨ ¬ w}
{r ∨ t, w ∨ ¬ r ∨ ¬ u, ¬ r ∨ ¬ u ∨ ¬ w}
•r•
{{r ∨ t}, {w ∨ ¬ r ∨ ¬ u, ¬ r ∨ ¬ u ∨ ¬ w}} → {t ∨ w ∨ ¬ u, t ∨ ¬ u ∨ ¬ w}
{t ∨ w ∨ ¬ u, t ∨ ¬ u ∨ ¬ w}
•t → 1•

{}
••{t → 1}••
{s ∨ ¬ p, w ∨ ¬ q, ¬ p ∨ ¬ u, ¬ q ∨ ¬ w, p ∨ q ∨ ¬ r}
•r → 0•
{s ∨ ¬ p, w ∨ ¬ q, ¬ p ∨ ¬ u, ¬ q ∨ ¬ w}
•w•
{{w ∨ ¬ q}, {¬ q ∨ ¬ w}} → {¬ q} , {s ∨ ¬ p, ¬ p ∨ ¬ u}

{¬ q, s ∨ ¬ p, ¬ p ∨ ¬ u}
•s → 1•
{¬ q, ¬ p ∨ ¬ u}
•u → 0•
{¬ q}
•q → 0•
{}
••{q → 0, r → 0, s → 1, t → 1, u → 0}••

Example 3

```
M2 = {p∨s, p∨r∨t, q∨r, q∨s, q∨¬p∨t, r∨¬q∨t, s∨¬r, t∨¬s, ¬t};
ResolutionSequence[M2, Reduce → True, Rules → True]
```

{p∨s, p∨r∨t, q∨r, q∨s, q∨t∨¬p, r∨t∨¬q, s∨¬r, t∨¬s, ¬t}
•p•
{{p∨s, p∨r∨t}, {q∨t∨¬p}} → {}, {q∨r, q∨s, r∨t∨¬q, s∨¬r, t∨¬s, ¬t}
{q∨r, q∨s, r∨t∨¬q, s∨¬r, t∨¬s, ¬t}
•q•
{{q∨r, q∨s}, {r∨t∨¬q}} → {r∨t, r∨s∨t}, {s∨¬r, t∨¬s, ¬t}
{r∨t, r∨s∨t, s∨¬r, t∨¬s, ¬t}
•r•
{{r∨t, r∨s∨t}, {s∨¬r}} → {s∨t}, {t∨¬s, ¬t}
{s∨t, t∨¬s, ¬t}
•s•
{{s∨t}, {t∨¬s}} → {t}, {¬t}
{t, ¬t}
•t•
{{t}, {¬t}} → {FALSE}
{FALSE}

```
M2 = {p∨s, p∨r∨t, q∨r, q∨s, q∨¬p∨t, r∨¬q∨t, s∨¬r, t∨¬s, ¬t};
ResolutionSequence[M2, Reduce → True, Rules → False]
```

{p∨s, p∨r∨t, q∨r, q∨s, q∨t∨¬p, r∨t∨¬q, s∨¬r, t∨¬s, ¬t}
•p•
{{p∨s, p∨r∨t}, {q∨t∨¬p}} → {}, {q∨r, q∨s, r∨t∨¬q, s∨¬r, t∨¬s, ¬t}
{q∨r, q∨s, r∨t∨¬q, s∨¬r, t∨¬s, ¬t}
•q•
{{q∨r, q∨s}, {r∨t∨¬q}} → {r∨t, r∨s∨t}, {s∨¬r, t∨¬s, ¬t}
{r∨t, r∨s∨t, s∨¬r, t∨¬s, ¬t}
•r•
{{r∨t, r∨s∨t}, {s∨¬r}} → {s∨t}, {t∨¬s, ¬t}
{s∨t, t∨¬s, ¬t}
•s•
{{s∨t}, {t∨¬s}} → {t}, {¬t}
{t, ¬t}
•t•
{{t}, {¬t}} → {FALSE}
{FALSE}

```
M2 = {p∨s, p∨r∨t, q∨r, q∨s, q∨¬p∨t, r∨¬q∨t, s∨¬r, t∨¬s, ¬t};
ResolutionSequence[M2, Reduce → False, Rules → True]
```

{p∨s, p∨r∨t, q∨r, q∨s, q∨t∨¬p, r∨t∨¬q, s∨¬r, t∨¬s, ¬t}
•p•
{{p∨s, p∨r∨t}, {q∨t∨¬p}} → {q∨s∨t, q∨r∨t} , {q∨r, q∨s, r∨t∨¬q, s∨¬r, t∨¬s, ¬t}
{q∨s∨t, q∨r∨t, q∨r, q∨s, r∨t∨¬q, s∨¬r, t∨¬s, ¬t}
•q•
{{q∨s∨t, q∨r∨t, q∨r, q∨s}, {r∨t∨¬q}} → {r∨s∨t, r∨t} , {s∨¬r, t∨¬s, ¬t}
{r∨s∨t, r∨t, s∨¬r, t∨¬s, ¬t}
•r•
{{r∨s∨t, r∨t}, {s∨¬r}} → {s∨t} , {t∨¬s, ¬t}
{s∨t, t∨¬s, ¬t}
•s•
{{s∨t}, {t∨¬s}} → {t} , {¬t}
{t, ¬t}
•t•
{{t}, {¬t}} → {FALSE}
{FALSE}

```
M2 = {p∨s, p∨r∨t, q∨r, q∨s, q∨¬p∨t, r∨¬q∨t, s∨¬r, t∨¬s, ¬t};
ResolutionSequence[M2, Reduce → False, Rules → False]
```

{p∨s, p∨r∨t, q∨r, q∨s, q∨t∨¬p, r∨t∨¬q, s∨¬r, t∨¬s, ¬t}
•p•
{{p∨s, p∨r∨t}, {q∨t∨¬p}} → {q∨s∨t, q∨r∨t} , {q∨r, q∨s, r∨t∨¬q, s∨¬r, t∨¬s, ¬t}
{q∨s∨t, q∨r∨t, q∨r, q∨s, r∨t∨¬q, s∨¬r, t∨¬s, ¬t}
•q•
{{q∨s∨t, q∨r∨t, q∨r, q∨s}, {r∨t∨¬q}} → {r∨s∨t, r∨t} , {s∨¬r, t∨¬s, ¬t}
{r∨s∨t, r∨t, s∨¬r, t∨¬s, ¬t}
•r•
{{r∨s∨t, r∨t}, {s∨¬r}} → {s∨t} , {t∨¬s, ¬t}
{s∨t, t∨¬s, ¬t}
•s•
{{s∨t}, {t∨¬s}} → {t} , {¬t}
{t, ¬t}
•t•
{{t}, {¬t}} → {FALSE}
{FALSE}

Example 4

```
M2 = {p∨s, p∨r∨t, q∨r, q∨s, q∨¬p∨t, r∨¬q∨t, s∨¬r, t∨¬s, ¬t};
ResolutionSequence[M2, Reduce → True, Rules → True, SelectionRule → Last]
```

{p∨s, p∨r∨t, q∨r, q∨s, q∨t∨¬p, r∨t∨¬q, s∨¬r, t∨¬s, ¬t}
•t → 0•
{p∨s, p∨r, q∨r, q∨s, q∨¬p, r∨¬q, s∨¬r, ¬s}
•s → 0•
{p, p∨r, q∨r, q, q∨¬p, r∨¬q, ¬r}
•r → 0•
{p, q, q∨¬p, ¬q}
•q•
{{q, q∨¬p}, {¬q}} → {FALSE, ¬p} , {p}
{FALSE, ¬p, p}

```
M2 = {p∨s, p∨r∨t, q∨r, q∨s, q∨¬p∨t, r∨¬q∨t, s∨¬r, t∨¬s, ¬t};
ResolutionSequence[M2, Reduce → True, Rules → False, SelectionRule → Random]
```

{p∨s, p∨r∨t, q∨r, q∨s, q∨t∨¬p, r∨t∨¬q, s∨¬r, t∨¬s, ¬t}
•t•
{{p∨r∨t, q∨t∨¬p, r∨t∨¬q, t∨¬s}, {¬t}} → {p∨r, q∨¬p, r∨¬q, ¬s} , {p∨s, q∨r, q∨s, s∨¬r}
{p∨r, q∨¬p, r∨¬q, ¬s, p∨s, q∨r, q∨s, s∨¬r}
•r•
{{p∨r, r∨¬q, q∨r}, {s∨¬r}} → {s∨¬q} , {q∨¬p, ¬s, p∨s, q∨s}
{s∨¬q, q∨¬p, ¬s, p∨s, q∨s}
•q•
{{q∨¬p, q∨s}, {s∨¬q}} → {s∨¬p, s} , {¬s, p∨s}
{s∨¬p, s, ¬s, p∨s}
•s•
{{s∨¬p, s, p∨s}, {¬s}} → {¬p, FALSE, p}
{¬p, FALSE, p}

```
M2 = {p∨s, p∨r∨t, q∨r, q∨s, q∨¬p∨t, r∨¬q∨t, s∨¬r, t∨¬s, ¬t};
ResolutionSequence[M2, Reduce → False, Rules → True, SelectionRule → Random]
```

{p∨s, p∨r∨t, q∨r, q∨s, q∨t∨¬p, r∨t∨¬q, s∨¬r, t∨¬s, ¬t}
•r•
{{p∨r∨t, q∨r, r∨t∨¬q}, {s∨¬r}} → {p∨s∨t, s∨t∨¬q} , {p∨s, q∨s, q∨t∨¬p, t∨¬s, ¬t}
{p∨s∨t, s∨t∨¬q, p∨s, q∨s, q∨t∨¬p, t∨¬s, ¬t}
•p•
{{p∨s∨t, p∨s}, {q∨t∨¬p}} → {q∨s∨t} , {s∨t∨¬q, q∨s, t∨¬s, ¬t}
{q∨s∨t, s∨t∨¬q, q∨s, t∨¬s, ¬t}
•t → 0•

{q ∨ s, s ∨ ¬ q, ¬ s}
•q•
{{q ∨ s}, {s ∨ ¬ q}} → {s} , {¬ s}
{s, ¬ s}
•s•
{{s}, {¬ s}} → {FALSE}
{FALSE}

```
M2 = {p ∨ s, p ∨ r ∨ t, q ∨ r, q ∨ s, q ∨ ¬ p ∨ t, r ∨ ¬ q ∨ t, s ∨ ¬ r, t ∨ ¬ s, ¬ t};
ResolutionSequence[M2, Reduce → False, Rules → False, Variables → {t, s, r, q, p}]
```

{p ∨ s, p ∨ r ∨ t, q ∨ r, q ∨ s, q ∨ t ∨ ¬ p, r ∨ t ∨ ¬ q, s ∨ ¬ r, t ∨ ¬ s, ¬ t}
•t•
{{p ∨ r ∨ t, q ∨ t ∨ ¬ p, r ∨ t ∨ ¬ q, t ∨ ¬ s}, {¬ t}} → {p ∨ r, q ∨ ¬ p, r ∨ ¬ q, ¬ s} , {p ∨ s, q ∨ r, q ∨ s, s ∨ ¬ r}
{p ∨ r, q ∨ ¬ p, r ∨ ¬ q, ¬ s, p ∨ s, q ∨ r, q ∨ s, s ∨ ¬ r}
•s•
{{p ∨ s, q ∨ s, s ∨ ¬ r}, {¬ s}} → {p, q, ¬ r} , {p ∨ r, q ∨ ¬ p, r ∨ ¬ q, q ∨ r}
{p, q, ¬ r, p ∨ r, q ∨ ¬ p, r ∨ ¬ q, q ∨ r}
•r•
{{p ∨ r, r ∨ ¬ q, q ∨ r}, {¬ r}} → {¬ q} , {p, q, q ∨ ¬ p}
{¬ q, p, q, q ∨ ¬ p}
•q•
{{q, q ∨ ¬ p}, {¬ q}} → {FALSE, ¬ p} , {p}
{FALSE, ¬ p, p}

ResolutionTable

ResolutionTable[x_List, options] tests whether the list x of clauses is satisfiable or not. It uses almost the same algorithm as ResolutionSequence but the result is presented in the form of a table.
Alternatives for options (default value = the first alternative) :
- BooleanValues -> {0, 1} | {False -> FalseSymbol, True -> TrueSymbol} | None,
- Dividers -> All | as for Grid,
- ItemSize -> Automatic | {{Automatic, {1.5}}, 1},
- Print -> False | True,
- Reduce -> False | True,
- Rules -> False | True,
- SelectionRule -> First | Last | Random,
- Sort -> DeleteDuplicates | Union,
- TestResult -> False | True,
- TableBreaks -> None | integer > 1 | increasing list of integers > 1,
- Transpose -> False | True,
- Variables -> All | List of logical variables.

```
Options[ResolutionTable]
```

```
{BooleanValues → {0, 1}, Dividers → All, ItemSize → Automatic, Print → False,
 Reduce → False, Rules → False, SelectionRule → First, Sort → DeleteDuplicates,
 TableBreaks → None, TestResult → False, Transpose → False, Variables → All}
```

Example 1

```
M1 = {p ∨ q ∨ ¬ r, ¬ p ∨ s ∨ ¬ t, r ∨ t, ¬ s ∨ t, ¬ p ∨ ¬ u, ¬ q ∨ w, ¬ q ∨ ¬ w};
ResolutionTable[M1, Reduce → False, Rules → False, TestResult → False]
```

	p	q	r	s	t	u	w
p ∨ q ∨ ¬ r	1	•	•	•	•	•	•
s ∨ ¬ p ∨ ¬ t	0	•	•	•	•	•	•
r ∨ t	•	•	1	•	•	•	•
t ∨ ¬ s	•	•	•	0	•	•	•
¬ p ∨ ¬ u	0	•	•	•	•	•	•
w ∨ ¬ q	•	0	•	•	•	•	•
¬ q ∨ ¬ w	•	0	•	•	•	•	•
•p•	…	…	…	…	…	…	…
q ∨ s ∨ ¬ r ∨ ¬ t	•	1	•	•	•	•	•
q ∨ ¬ r ∨ ¬ u	•	1	•	•	•	•	•
•q•	…	…	…	…	…	…	…
s ∨ w ∨ ¬ r ∨ ¬ t	•	•	0	•	•	•	•
w ∨ ¬ r ∨ ¬ u	•	•	0	•	•	•	•
s ∨ ¬ r ∨ ¬ t ∨ ¬ w	•	•	0	•	•	•	•
¬ r ∨ ¬ u ∨ ¬ w	•	•	0	•	•	•	•
•r•	…	…	…	…	…	…	…
t ∨ w ∨ ¬ u	•	•	•	•	1	•	•
t ∨ ¬ u ∨ ¬ w	•	•	•	•	1	•	•
•s•	…	…	…	…	…	…	…
•t•	…	…	…	…	…	…	…
•u•	…	…	…	…	…	…	…
•w•	…	…	…	…	…	…	…

```
M1 = {p ∨ q ∨ ¬ r, ¬ p ∨ s ∨ ¬ t, r ∨ t, ¬ s ∨ t, ¬ p ∨ ¬ u, ¬ q ∨ w, ¬ q ∨ ¬ w};
{ResolutionTable[M1, Reduce → True, Rules → True, SelectionRule → First, TestResult → True],
  "   ", ResolutionTable[M1, Reduce → True, Rules → True,
   SelectionRule → First, TestResult → False]} // Row
```

■	p	q	r	s	t	u	w
p ∨ q ∨ ¬ r	1	•	•	•	•	•	•
s ∨ ¬ p ∨ ¬ t	0	•	•	•	•	•	•
r ∨ t	•	•	1	•	•	•	•
t ∨ ¬ s	•	•	•	0	•	•	•
¬ p ∨ ¬ u	0	•	•	•	•	•	•
w ∨ ¬ q	•	0	•	•	•	•	•
¬ q ∨ ¬ w	•	0	•	•	•	•	•
•p•	…	…	…	…	…	…	…
q ∨ s ∨ ¬ r ∨ ¬ t	•	1	•	•	•	•	•
q ∨ ¬ r ∨ ¬ u	•	1	•	•	•	•	•
•q•	…	…	…	…	…	…	…
s ∨ w ∨ ¬ r ∨ ¬ t	•	•	0	•	•	•	•
w ∨ ¬ r ∨ ¬ u	•	•	0	•	•	•	•
s ∨ ¬ r ∨ ¬ t ∨ ¬ w	•	•	0	•	•	•	•
¬ r ∨ ¬ u ∨ ¬ w	•	•	0	•	•	•	•
•r•	…	…	…	…	…	…	…
t ∨ w ∨ ¬ u	•	•	•	•	1	•	•
t ∨ ¬ u ∨ ¬ w	•	•	•	•	1	•	•
•s → 0•	…	…	…	…	…	…	…
•t → 1•	…	…	…	…	…	…	…
■	p	q	r	0	1	u	w
□	0	q	r	0	1	u	w
□	0	q	0	0	1	u	w
□	0	0	0	0	1	u	w

■	p	q	r	s	t	u	w
p ∨ q ∨ ¬ r	1	•	•	•	•	•	•
s ∨ ¬ p ∨ ¬ t	0	•	•	•	•	•	•
r ∨ t	•	•	1	•	•	•	•
t ∨ ¬ s	•	•	•	0	•	•	•
¬ p ∨ ¬ u	0	•	•	•	•	•	•
w ∨ ¬ q	•	0	•	•	•	•	•
¬ q ∨ ¬ w	•	0	•	•	•	•	•
•p•	…	…	…	…	…	…	…
q ∨ s ∨ ¬ r ∨ ¬ t	•	1	•	•	•	•	•
q ∨ ¬ r ∨ ¬ u	•	1	•	•	•	•	•
•q•	…	…	…	…	…	…	…
s ∨ w ∨ ¬ r ∨ ¬ t	•	•	0	•	•	•	•
w ∨ ¬ r ∨ ¬ u	•	•	0	•	•	•	•
s ∨ ¬ r ∨ ¬ t ∨ ¬ w	•	•	0	•	•	•	•
¬ r ∨ ¬ u ∨ ¬ w	•	•	0	•	•	•	•
•r•	…	…	…	…	…	…	…
t ∨ w ∨ ¬ u	•	•	•	•	1	•	•
t ∨ ¬ u ∨ ¬ w	•	•	•	•	1	•	•
•s → 0•	…	…	…	…	…	…	…
•t → 1•	…	…	…	…	…	…	…

True

Example 2

```
M2 = {q ∨ ¬ p ∨ t, r ∨ ¬ q ∨ t, s ∨ ¬ r, p ∨ r ∨ t, t ∨ ¬ s, q ∨ ¬ p, ¬ t};
ResolutionTable[M2, ItemSize → {{7, {1}}, 1}, Print → False,
  Reduce → False, Rules → True, Sort → DeleteDuplicates, TableBreaks → 10,
  TestResult → True, Variables → {t, s, r, q, p}] // Row[#, ",   "] &
```

■	t	s	r	q	p
r ∨ t ∨ ¬ q	1	•	•	•	•
s ∨ ¬ r	•	1	•	•	•
p ∨ r ∨ t	1	•	•	•	•
t ∨ ¬ s	1	•	•	•	•
q ∨ ¬ p	•	•	•	1	•
¬ t	0	•	•	•	•
•t → 0•	…	…	…	…	…
r ∨ ¬ q	•	•	1	•	•
p ∨ r	•	•	1	•	•
¬ s	•	0	•	•	•

,

■	t	s	r	q	p
•s → 0•	…	…	…	…	…
¬ r	•	•	0	•	•
•r → 0•	…	…	…	…	…
¬ q	•	•	•	0	•
p	•	•	•	•	1
•q → 0•	…	…	…	…	…
¬ p	•	•	•	•	0
•p•	…	…	…	…	…
FALSE	•	•	•	•	•

```
M2 = {q ∨ ¬ p ∨ t, r ∨ ¬ q ∨ t, s ∨ ¬ r, p ∨ r ∨ t, t ∨ ¬ s, q ∨ ¬ p, ¬ t};
ResolutionTable[M2, ItemSize → {Automatic, {2, {1}}}, Reduce → False, Rules → True,
   Sort → DeleteDuplicates, TableBreaks → 5, Transpose → True, Variables → {t, s, q, r, p}] //
  Partition[#, 2] & // Grid[#, Alignment → {Left, Top}] &
```

■	r ∨ t ∨ ¬ q	s ∨ ¬ r	p ∨ r ∨ t	t ∨ ¬ s	q ∨ ¬ p
t	1	•	1	1	•
s	•	1	•	•	•
q	•	•	•	•	1
r	•	•	•	•	•
p	•	•	•	•	•

■	¬ t	•t → 0•	r ∨ ¬ q	p ∨ r	¬ s
t	0	…	•	•	•
s	•	…	•	•	0
q	•	…	0	•	•
r	•	…	•	1	•
p	•	…	•	•	•

■	•s → 0•	¬ r	•q•	r ∨ ¬ p	•r → 0•
t	…	•	…	•	…
s	…	•	…	•	…
q	…	•	…	•	…
r	…	0	…	1	…
p	…	•	…	•	…

■	¬ p	p	•p•	FALSE
t	•	•	…	•
s	•	•	…	•
q	•	•	…	•
r	•	•	…	•
p	0	1	…	•

RefutationTree

RefutationTree[x] decides whether the list x of clauses is satisfiable or not and in the later case outputs refutation tree of a list of in a convenient graphic form.
Alternatives for Options (default value = the first alternative):
- Format -> TreePlot | TableForm | List,
- Frame -> True | False,
- RootPosition -> Right | Left | Top | Bottom,
- SelectionRule -> First | Last | Random,
- Sort -> DeleteDuplicates | Union,
- TableSpacing -> Automatic | {nonnegative integer, nonnegative integer}.

```
Options[RefutationTree]
```

```
{AspectRatio → Automatic, DirectedEdges → True, Format → TreePlot,
 Frame → True, ImageSize → {500, Automatic}, ImagePadding → All,
 PlotTheme → ClassicLabeled, RootPosition → Right, Root → FALSE,
 SelectionRule → First, Sort → DeleteDuplicates, TableSpacing → {0, 0}}
```

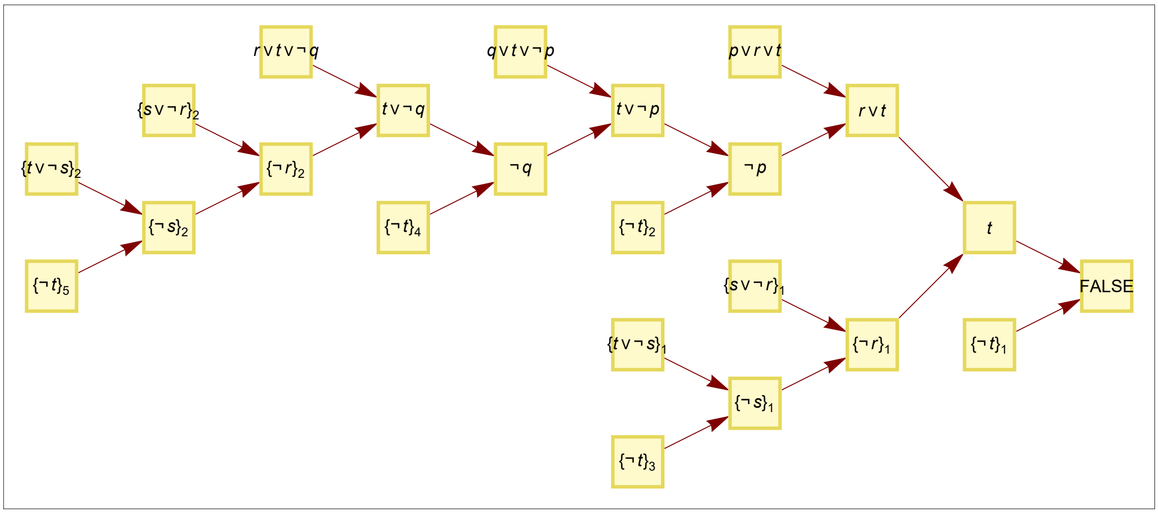
Example 1

```
M1 = {p ∨ q ∨ ¬ r, ¬ p ∨ s ∨ ¬ t, r ∨ t, ¬ s ∨ t, ¬ p ∨ ¬ u, ¬ q ∨ w, ¬ q ∨ ¬ w};
RefutationTree[M1, PlotTheme → "ClassicLabeled"]
```

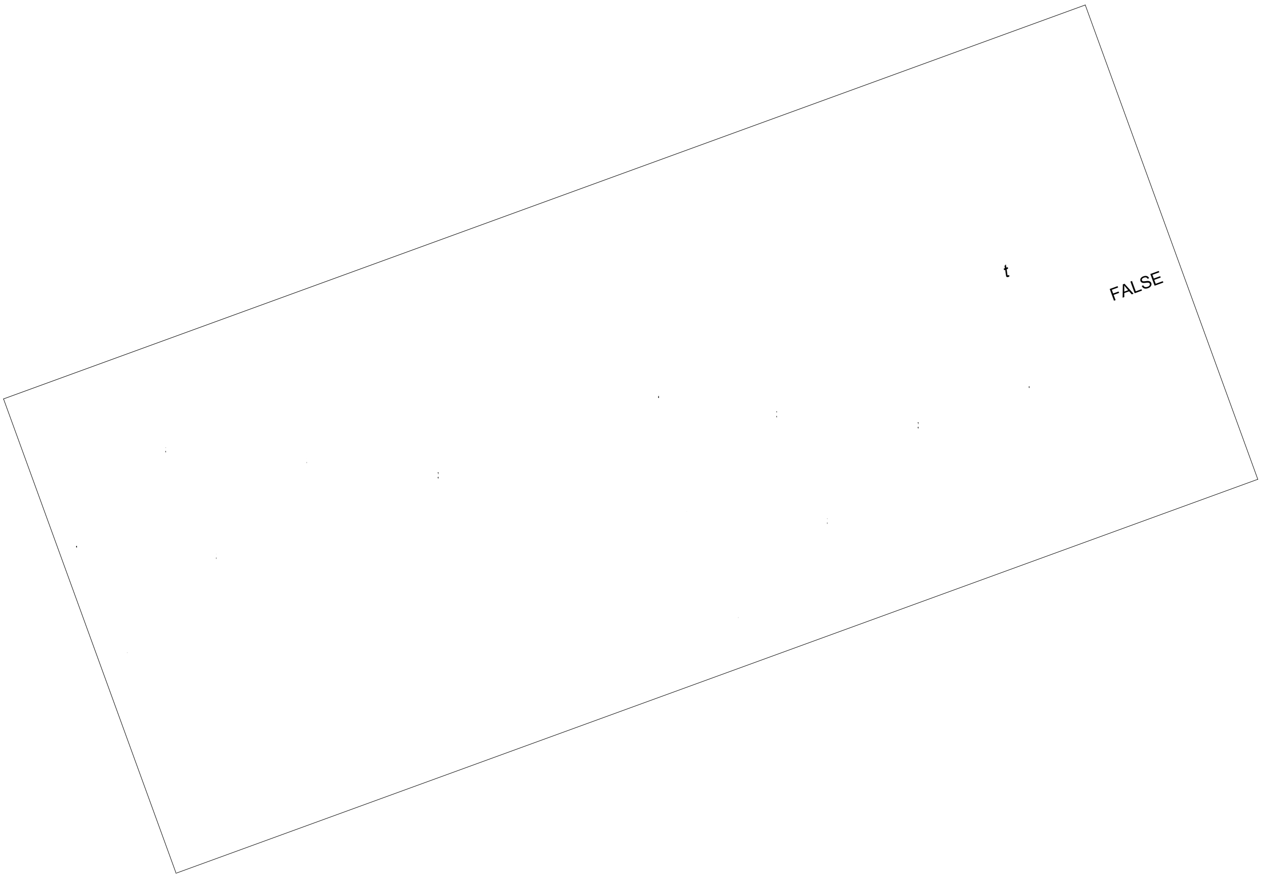
```
The set of clauses is satisfiable
```

Example 2

```
M2 = {q ∨ ¬ p ∨ t, r ∨ ¬ q ∨ t, s ∨ ¬ r, p ∨ r ∨ t, t ∨ ¬ s, q ∨ ¬ p, ¬ t};
tree = RefutationTree[M2, ImageSize → {600, Automatic}, PlotTheme → "ClassicLabeled"]
```

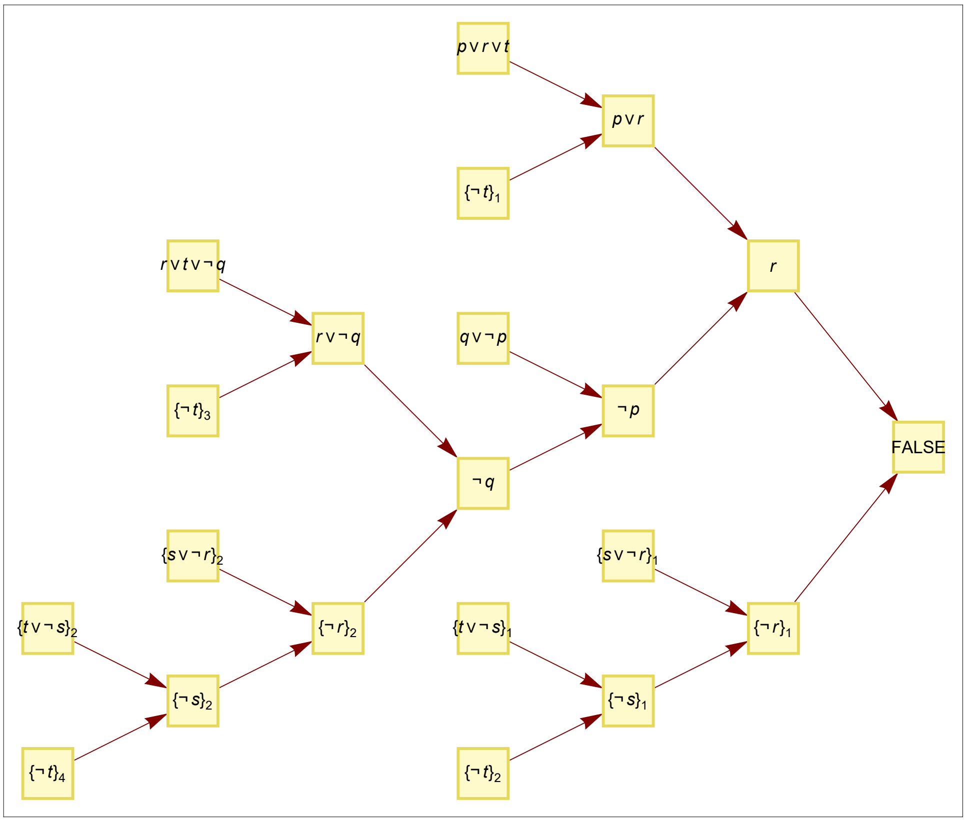



```
Rotate[tree, 20 Degree]
```



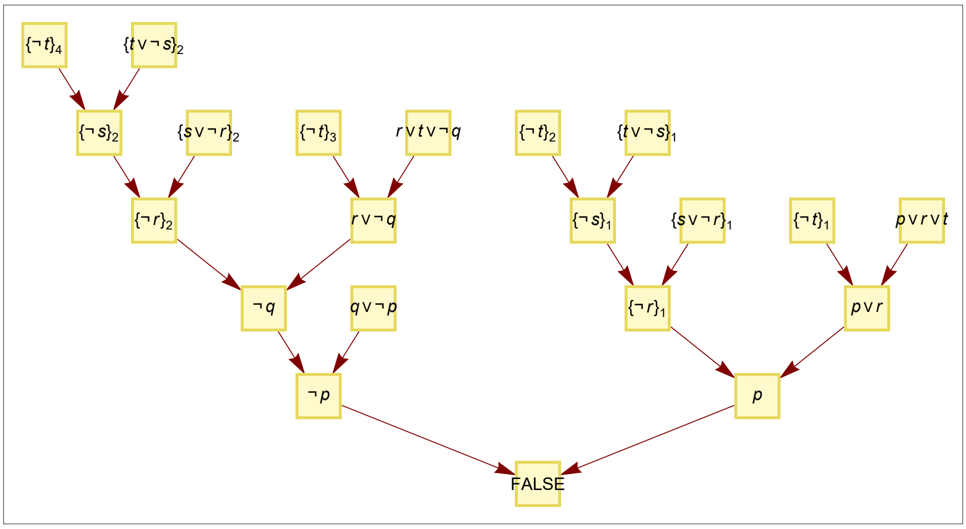
Example 3

```
M2 = {q ∨ ¬ p ∨ t, r ∨ ¬ q ∨ t, s ∨ ¬ r, p ∨ r ∨ t, t ∨ ¬ s, q ∨ ¬ p, ¬ t};
RefutationTree[M2, RootPosition → Right,
 SelectionRule → Random, PlotTheme → "ClassicLabeled", VertexSize → 0.8]
```



Example 4

```
M2 = {q ∨ ¬ p ∨ t, r ∨ ¬ q ∨ t, s ∨ ¬ r, p ∨ r ∨ t, t ∨ ¬ s, q ∨ ¬ p, ¬ t};
gg = RefutationTree[M2, AspectRatio → 0.5,
  PlotTheme → "ClassicLabeled", RootPosition → Bottom, SelectionRule → Last]
```



Example 5

```
M2 = {q ∨ ¬ p ∨ t, r ∨ ¬ q ∨ t, s ∨ ¬ r, p ∨ r ∨ t, t ∨ ¬ s, q ∨ ¬ p, ¬ t};
RefutationTree[M2, Format → TableForm]
```

```
                                      p ∨ r ∨ t
                                                 •r ∨ t•
                            q ∨ t ∨ ¬ p
                                        •t ∨ ¬ p•
                 r ∨ t ∨ ¬ q
                              •t ∨ ¬ q•
             s ∨ ¬ r
                     •¬ r•
      t ∨ ¬ s
              •¬ s•
         ¬ t
                                •¬ q•
                           ¬ t
                                           •¬ p•
                                      ¬ t
                                                            •t•
                                          s ∨ ¬ r
                                                   •¬ r•
                                     t ∨ ¬ s
                                             •¬ s•
                                        ¬ t
                                                                 •FALSE•
                                                            ¬ t
```

```
M2 = {q ∨ ¬ p ∨ t, r ∨ ¬ q ∨ t, s ∨ ¬ r, p ∨ r ∨ t, t ∨ ¬ s, q ∨ ¬ p, ¬ t};
RefutationTree[M2, Format → TableForm, RootPosition → Left, TableSpacing → {0.5, 1}]
```

```
                      p ∨ r ∨ t
             •r ∨ t•
                                      q ∨ t ∨ ¬ p
                    •t ∨ ¬ p•
                                                    r ∨ t ∨ ¬ q
                              •t ∨ ¬ q•
                                                          s ∨ ¬ r
                                     •¬ r•
                                                                   t ∨ ¬ s
                                           •¬ s•
                                                                   ¬ t
                                 •¬ q•
                                        ¬ t
                     •¬ p•
                              ¬ t
        •t•
                     s ∨ ¬ r
             •¬ r•
                              t ∨ ¬ s
                    •¬ s•
                              ¬ t
•FALSE•
         ¬ t
```

Example 6

```
M2 = {q ∨ ¬ p ∨ t, r ∨ ¬ q ∨ t, s ∨ ¬ r, p ∨ r ∨ t, t ∨ ¬ s, q ∨ ¬ p, ¬ t};
RefutationTree[M2, Format -> List]
```

{t → FALSE, r ∨ t → t, p ∨ r ∨ t → r ∨ t, ¬ p → r ∨ t, t ∨ ¬ p → ¬ p, q ∨ t ∨ ¬ p → t ∨ ¬ p,
¬ q → t ∨ ¬ p, t ∨ ¬ q → ¬ q, r ∨ t ∨ ¬ q → t ∨ ¬ q, $\{\neg r\}_2$ → t ∨ ¬ q, $\{s \vee \neg r\}_2 \to \{\neg r\}_2$,
$\{\neg s\}_2 \to \{\neg r\}_2$, $\{t \vee \neg s\}_2 \to \{\neg s\}_2$, $\{\neg t\}_5 \to \{\neg s\}_2$, $\{\neg t\}_4 \to \neg q$, $\{\neg t\}_2 \to \neg p$, $\{\neg r\}_1 \to t$,
$\{s \vee \neg r\}_1 \to \{\neg r\}_1$, $\{\neg s\}_1 \to \{\neg r\}_1$, $\{t \vee \neg s\}_1 \to \{\neg s\}_1$, $\{\neg t\}_3 \to \{\neg s\}_1$, $\{\neg t\}_1 \to$ FALSE}

```
graph = RefutationTree[M2, Format -> List] /. {Subscript[{u_}, _] → u} // Union
```

{t → FALSE, ¬ p → r ∨ t, ¬ q → t ∨ ¬ p, ¬ r → t, ¬ r → t ∨ ¬ q, ¬ s → ¬ r,
¬ t → FALSE, ¬ t → ¬ p, ¬ t → ¬ q, ¬ t → ¬ s, r ∨ t → t, s ∨ ¬ r → ¬ r, t ∨ ¬ p → ¬ p,
t ∨ ¬ q → ¬ q, t ∨ ¬ s → ¬ s, p ∨ r ∨ t → r ∨ t, q ∨ t ∨ ¬ p → t ∨ ¬ p, r ∨ t ∨ ¬ q → t ∨ ¬ q}

```
Framed[Graph[graph, AspectRatio → 1 / 2, GraphLayout → "SpringElectricalEmbedding",
  ImageSize → {600, 350}, ImagePadding → All, ImageMargins → 0,
  VertexLabels → Placed["Name", {Above, After}], VertexLabelStyle → Directive[Red, Bold, 14],
  VertexShapeFunction → "Circle", VertexSize → 0.03 {1, 2}]]
```