

Satisfiable Boolean Expressions

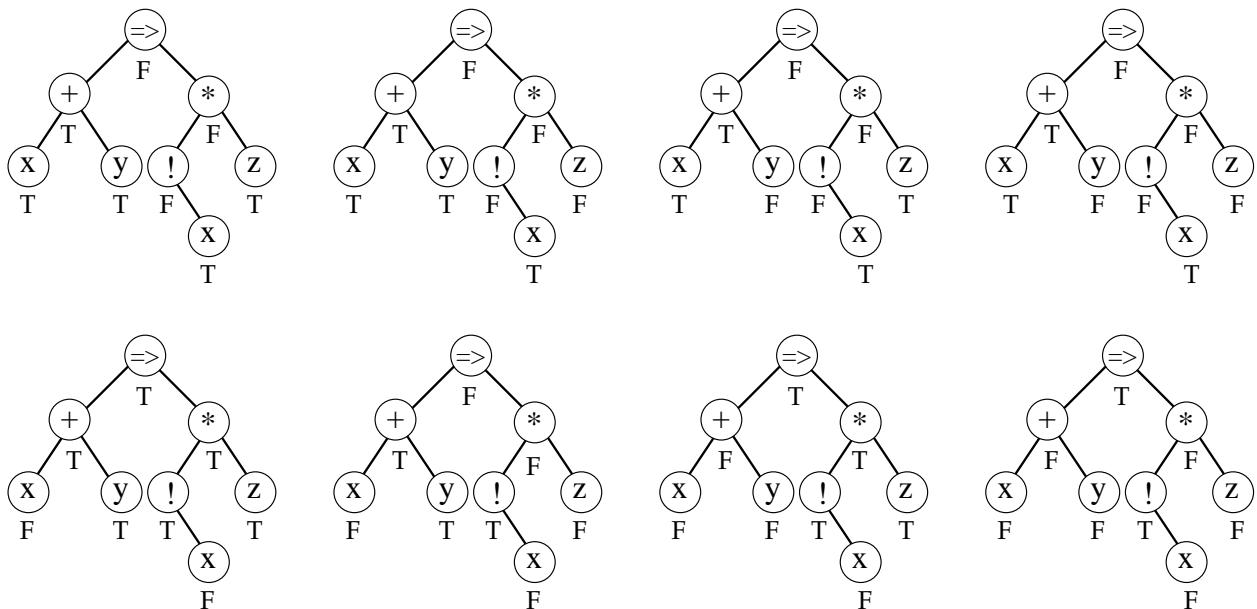
A Boolean expression may have parentheses and the following Boolean operators. All variables within a Boolean expression are presumed to be of type Boolean.

A Boolean expression is *satisfiable* if there is some assignment of truth values to its variables such that the value of the expression is true. Otherwise, it is a *contradiction*. The expression is a *tautology* if its value is true for every assignment.

1. $x + y$ meaning x or y .
2. $x * y$ meaning x and y .
3. $!x$ meaning not x .
4. $x \Rightarrow y$ meaning x implies y .

Note that $x \Rightarrow y$ is equivalent to $!x + y$. Each Boolean expression has an *expression tree*, where the internal nodes are operators and the leaves are operands. See <https://www.geeksforgeeks.org/expression-tree/>

If an operand is a variable, it can be assigned one of two values, T and F . Given an assignment of its variables, the truth value of a Boolean expression is computed bottom-up.



The above figures show the expression trees for the Boolean expression $(x + y) \Rightarrow (!x * z)$, one for each of the eight possible assignments. The expression has three satisfying assignments, and thus satisfiable. However, it is not a tautology, since not every assignment is satisfying.

Truth Tables

The truth table for a Boolean expression gives the truth value of the expression for every possible assignment. There is one row of the table for each assignment of the variables of the expression. Below, we show the bottom-up construction of the truth table for the expression $(x + y) \Rightarrow (!x * z)$.

$$(x + y) \Rightarrow (!x * z)$$

<i>T</i>	<i>T</i>	<i>T</i>	<i>T</i>
<i>T</i>	<i>T</i>	<i>T</i>	<i>F</i>
<i>T</i>	<i>F</i>	<i>T</i>	<i>T</i>
<i>T</i>	<i>F</i>	<i>T</i>	<i>F</i>
<i>F</i>	<i>T</i>	<i>F</i>	<i>T</i>
<i>F</i>	<i>T</i>	<i>F</i>	<i>F</i>
<i>F</i>	<i>F</i>	<i>F</i>	<i>T</i>
<i>F</i>	<i>F</i>	<i>F</i>	<i>F</i>

$$(x + y) \Rightarrow (!x * z)$$

<i>T</i>	<i>T</i>	<i>T</i>	<i>F</i>	<i>T</i>	<i>T</i>
<i>T</i>	<i>T</i>	<i>T</i>	<i>F</i>	<i>T</i>	<i>F</i>
<i>T</i>	<i>T</i>	<i>F</i>	<i>F</i>	<i>T</i>	<i>T</i>
<i>T</i>	<i>T</i>	<i>F</i>	<i>F</i>	<i>T</i>	<i>F</i>
<i>F</i>	<i>T</i>	<i>T</i>	<i>T</i>	<i>F</i>	<i>T</i>
<i>F</i>	<i>T</i>	<i>T</i>	<i>T</i>	<i>F</i>	<i>F</i>
<i>F</i>	<i>F</i>	<i>F</i>	<i>T</i>	<i>F</i>	<i>T</i>
<i>F</i>	<i>F</i>	<i>F</i>	<i>T</i>	<i>F</i>	<i>F</i>

$$(x + y) \Rightarrow (!x * z)$$

<i>T</i>	<i>T</i>	<i>T</i>	<i>F</i>	<i>T</i>	<i>F</i>	<i>T</i>
<i>T</i>	<i>T</i>	<i>T</i>	<i>F</i>	<i>T</i>	<i>F</i>	<i>F</i>
<i>T</i>	<i>T</i>	<i>F</i>	<i>F</i>	<i>T</i>	<i>F</i>	<i>T</i>
<i>T</i>	<i>T</i>	<i>F</i>	<i>F</i>	<i>T</i>	<i>F</i>	<i>F</i>
<i>F</i>	<i>T</i>	<i>T</i>	<i>T</i>	<i>F</i>	<i>T</i>	<i>T</i>
<i>F</i>	<i>T</i>	<i>T</i>	<i>T</i>	<i>F</i>	<i>F</i>	<i>F</i>
<i>F</i>	<i>F</i>	<i>F</i>	<i>T</i>	<i>F</i>	<i>T</i>	<i>T</i>
<i>F</i>	<i>F</i>	<i>F</i>	<i>T</i>	<i>F</i>	<i>F</i>	<i>F</i>

$$(x + y) \Rightarrow (!x * z)$$

<i>T</i>	<i>T</i>	<i>T</i>	<i>F</i>	<i>F</i>	<i>T</i>	<i>F</i>	<i>T</i>
<i>T</i>	<i>T</i>	<i>T</i>	<i>F</i>	<i>F</i>	<i>T</i>	<i>F</i>	<i>F</i>
<i>T</i>	<i>T</i>	<i>F</i>	<i>F</i>	<i>F</i>	<i>T</i>	<i>F</i>	<i>T</i>
<i>T</i>	<i>T</i>	<i>F</i>	<i>F</i>	<i>F</i>	<i>T</i>	<i>F</i>	<i>F</i>
<i>F</i>	<i>T</i>	<i>T</i>	<i>T</i>	<i>T</i>	<i>F</i>	<i>T</i>	<i>T</i>
<i>F</i>	<i>T</i>	<i>T</i>	<i>F</i>	<i>T</i>	<i>F</i>	<i>F</i>	<i>F</i>
<i>F</i>	<i>F</i>	<i>F</i>	<i>T</i>	<i>T</i>	<i>F</i>	<i>T</i>	<i>T</i>
<i>F</i>	<i>F</i>	<i>F</i>	<i>T</i>	<i>T</i>	<i>F</i>	<i>F</i>	<i>F</i>