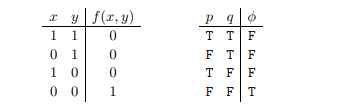
**Propositional formulas and truth tables**

Truth tables and propositional formulas are two different representations of boolean functions. In propositional formulas, ∧ denotes ·, ∨ denotes +, ¬ denotes ¯ and and ⊥ denote 1 and 0, respectively.

Boolean functions are represented by truth tables in the obvious way; for example, the function f(x, y) def = x + y is represented by the truth table on the left:



On the right, we show the same truth table using the notation. we may mix these two notational systems of boolean formulas and formulas of propositional logic whenever it is convenient. You should be able to translate expressions easily from one notation to the other and vice versa.

As representations of boolean functions, propositional formulas and truth tables have different advantages and disadvantages. Truth tables are very space-inefficient: if one wanted to model the functionality of a sequential circuit by a boolean function of 100 variables (a small chip component would easily require this many variables), then the truth table would require 2100 (which is more than 1030) lines. Alas, there is not enough storage space (whether paper or particle) in the universe to record the information of 2100 different bit vectors of length 100. Although they are space inefficient, operations on truth tables are simple. Once you have computed a truth table, it is easy to see whether the boolean function represented is satisfiable: you just look to see if there is a 1 in the last column of the table.

Comparing whether two truth tables represent the same boolean function also seems easy: assuming the two tables are presented with the same order of valuations, we simply check that they are identical. Although these operations seem simple, however, they are computationally intractable because of the fact that the number of lines in the truth table is exponential in the number of variables. Checking satisfiability of a function with n atoms requires of the order of 2n operations if the function is represented as a truth table. We conclude that checking satisfiability and equivalence is highly inefficient with the truth-table representation.

Representation of boolean functions by propositional formulas is slightly better. Propositional formulas often provide a wonderfully compact and efficient presentation of boolean functions. A formula with 100 variables might only be about 200–300 characters long. However, deciding whether an arbitrary propositional formula is satisfiable is a famous problem in computer science: no efficient algorithms for this task are known, and it is strongly suspected that there aren’t any. Similarly, deciding whether two arbitrary propositional formulas f and g denote the same boolean function is suspected to be exponentially expensive.