**Binary decision diagrams**

**Representing Boolean functions**

Boolean functions are an important descriptive formalism for many hardware and software systems, such as synchronous and asynchronous circuits, reactive systems and finite-state programs. Representing those systems in a computer in order to reason about them requires an efficient representation for boolean functions

A boolean variable x is a variable ranging over the values 0 and 1. We write x1, x2,... and x, y, z,... to denote boolean variables. We define the following functions on the set {0, 1}:

* 0 def = 1 and 1 def = 0;
* x · y def = 1 if x and y have value 1; otherwise x · y def = 0;
* x + y def = 0 if x and y have value 0; otherwise x + y def = 1;
* x ⊕ y def = 1 if exactly one of x and y equals 1.

A boolean function f of n arguments is a function from {0, 1}n to {0, 1}.

We write f(x1, x2,...,xn), or f(V ), to indicate that a syntactic representation of f depends on the boolean variables in V only.

Note that ·, + and ⊕ are boolean functions with two arguments, whereas ¯ is a boolean function that takes one argument. The binary functions ·, + and ⊕ are written in infix notation instead of prefix; i.e. we write x + y instead of +(x, y), etc.

In terms of the four functions above, we can define other boolean functions such as

(1) f(x, y) def = x · (y + x)

(2) g(x, y) def = x · y + (1 ⊕ x)

(3) h(x, y, z) def = x + y · (x ⊕ y)

(4) k() def = 1 ⊕ (0 · 1).