CS304 – CS310: Mathematical and Logical Aspects of Computing

Lecture 2: Boolean Expressions

Friday, 6th Sep 2013

Dog logic

TODAY:

1. Binary numbers and a half adder.
2. Boolean expressions; assignments and interpretations.
3. Logical equivalences; distributive laws.

References: Ben-Ari *Mathematical Logic for Computer Science*, particularly Sections 1.1, 1.2, 1.3; 2.1.

A Boolean operator takes arguments of *Boolean type* (i.e., with one of the true values F or T assigned to them) and maps these to one of F or T.

If the operator takes n arguments, we say it is an *n-place Boolean operator*.

- There are **4** possible 1-place operators, of which **negation** is the most interesting. It is denoted by $\neg$, and read as "not".
- There are **16** possible 2-place operators. Among the most important are
  - *Conjunction*, written as $a \wedge b$, read as "and".
  - *Disjunction*, written as $a \vee b$, read as "or".
  - *Non-equivalence*, a.k.a., the *exclusive or* operator, and denoted as $\oplus$

The formal study of logic dates from (at least) the ancient Greeks, and in particular, the work of Plato in the 4^{th} century, BC.

The formal study of logic gain impetuous in the 19th century (e.g., Boole, Venn, and de Morgan: three names which we'll hear often through this course). Much of the effort came from trying to establish if an axiomatic theory is

- consistent,
- independent,
- sound,
- complete.

The formalism that was introduced, e.g., by Boole, would later have very important applications in the design of electronic circuits. So, before we continue the mathematical study, we'll consider that application.

Digital electronics are based on the idea of presenting pieces of data as discrete values. The signals they operator on are discrete, and represent two voltage bands:

- zero volts (i.e., ground) which corresponds to the "false" (symbols: F, $\bot$, 0), and
- the supply voltage, which corresponds to "true" (symbols: T, $\top$, 1).

With this we are represent, for example, (base 10) numbers in binary: e.g.,

It is not enough to be able to store numbers, we have to be able to manipulate them too, e.g., by addition, subtractions, multiplication, etc...

As addition is the most fundamental operation, we'll see how to construct such a circuit using the Boolean operators: $\{\wedge, \vee, \neg\}$.

Our inputs are two binary digits, a and b, and the outputs are the sum, S, and carry, C.

EXERCISE: Show how to build a full adder. The inputs are bits a and b, and the carry from the previous sum C_1. The outputs are the sum, S, and the new carry, C_2.

The half-adder we constructed in class gave $C = a \wedge b$, and $S = (a \vee b) \wedge \neg(a \wedge b)$. But recalling the examples of Boolean operators from Lecture one, we realise that we could write $S = a \oplus b$.

That is, we see that these two Boolean operators, $a \oplus b$ and $(a \vee b) \wedge \neg(a \wedge b)$ are essentially the same. This leads us towards the idea of "logical equivalence".

Often we have used the symbols a and b as variables that could take the value F or T. Such variables are usually called "*Boolean variables*", "*propositional variables*", "*atomic propositions*" or, simply, "*atoms*".

We will denote this set of variables as $\mathcal{P}$. By writing $x \in \mathcal{P}$, we mean that x is a Boolean variable that can have the value F or T.

Definition (Boolean Expression / Propositional formula)

We can define the set of all **Boolean Expressions** (a.k.a. Formulas), denoted $\mathcal{F}$ the following, recursive manner:

1. F, T and $x_i \in \mathcal{P}$ are Boolean Expressions. That is, they belong to the set $\mathcal{F}$.
2. If E belongs to $\mathcal{F}$, then so too does $\neg E$.
3. If $\star$ is a 2-place Boolean operator, and E_1 and E_2 are both in $\mathcal{F}$, then then expression $E_1 \star E_2$ is in $\mathcal{F}$.

When we give a variable a particular value or F or T, we call this an "*assignment*". So every Boolean variable has two possible assignments.

When we give a particular assignment to arguments of a Boolean Expression, this is called an "*interpretation*". This means that every 2-place operator has **four** possible interpretations.

Examples:

. . . . . .

Definition (Logically equivalent)

Two Boolean expressions $E_1, E_2 \in \mathcal{F}$ are *logically equivalent* if $v(E_1) = v(E_2)$ for all possible interpretations v. We write this as $E_1 \equiv E_2$. Informally, they are logically equivalent if their truth-table are the same.

Now that we have this definition we can investigate some important Boolean identities. **Examples:**

Example

Use a logic table to show that

$$(a \vee b) \wedge c \text{ is logically equivalent to } (a \wedge c) \vee (b \wedge c).$$

. . . . . .

EXERCISE The above identity is an example of a *distributive law*. Determine if $(a \wedge b) \vee c$ is logically equivalent to $(a \vee c) \wedge (b \vee c)$.