ARITHMETIC BOOLEAN EXPRESSIONS

9.1 Introduction

Boolean expressions are sometimes used in the research and development of digital systems, but calculating Boolean expressions by hand is a cumbersome job, even when they have only a few variables. For example, the Boolean expressions

\[(a \land \overline{b}) \lor (\overline{a} \land \overline{c}) \lor (b \land c),\]
\[(a \land \overline{b}) \lor (a \land c) \lor (\overline{a} \land \overline{b}) \lor (\overline{a} \land \overline{c}),\]
\[and\ (a \land \overline{b}) \lor (\overline{a} \land b) \lor (b \land c) \lor (\overline{b} \land \overline{c})\]

represent the same function, but it is hard to verify their equivalence by hand. If they have more than five or six variables, we might as well give up. This problem motivated us to develop a Boolean expression manipulator (BEM) [MIY89], which is an interpreter that uses BDDs to calculate Boolean expressions. It enables us to check the equivalence and implications of Boolean expressions easily, and it helped us in developing VLSI design systems and solving combinatorial problems.

Most discrete problems can be described by logic expressions; however, the arithmetic operators such as addition, subtraction, multiplication, and comparison, are convenient for describing many practical problems. Such expressions can be rewritten using logic operators only, but this can result in expressions that are complicated and hard to read. In many cases, arithmetic operators provide simple descriptions of problems.

In this chapter, we present a new Boolean expression manipulator, that allows the use of arithmetic operators [Min93a]. This manipulator, called BEM-II, can directly solve problems represented by a set of equalities and inequalities, which are dealt with in 0-1 linear programming. Of course, it can also manipulate ordinary Boolean expressions. We developed several output formats for displaying expressions containing arithmetic operators.
In the following sections, we first show a method for manipulating Boolean expressions with arithmetic operations. We then present implementation of the BEM-II and its applications.

### 9.2 MANIPULATION OF ARITHMETIC BOOLEAN EXPRESSIONS

Most discrete problems can be described by using logic operations; however, arithmetic operators are useful for describing many practical problems. For example, a majority function with five inputs can be expressed concisely by using arithmetic operators:

\[ x_1 + x_2 + x_3 + x_4 + x_5 \geq 3. \]

When, on the other hand, it is written using only Boolean expressions, this function becomes more complicated:

\[
(x_1 \land x_2 \land x_3) \lor (x_1 \land x_2 \land x_4) \lor (x_1 \land x_2 \land x_5) \\
\lor (x_1 \land x_3 \land x_4) \lor (x_1 \land x_3 \land x_5) \lor (x_1 \land x_4 \land x_5) \\
\lor (x_2 \land x_3 \land x_4) \lor (x_2 \land x_3 \land x_5) \lor (x_2 \land x_4 \land x_5) \\
\lor (x_3 \land x_4 \land x_5). 
\]

In this section, we describe an efficient method for representing and manipulating Boolean expressions containing arithmetic operators.

#### 9.2.1 Definitions

We define arithmetic Boolean expressions and Boolean-to-integer functions, which are extended models of conventional Boolean expressions and Boolean functions.

**Definition 9.1** Arithmetic Boolean expressions are extended Boolean expressions that contain not only logic operators, but also arithmetic operators, such as addition (+), subtraction (−), and multiplication (×). Any integer can be used as a constant in the expression, but input variables are restricted to either 0 or 1. Equality (=) and inequalities (<, >, ≤, ≥, ≠) are defined as operations returning a value of either 1 (true) or 0 (false).

For example, \((3 \times x_1 + x_2)\) is an arithmetic Boolean expression with respect to the variables \(x_1, x_2 \in \{0, 1\}\). \((3 \times x_1 + x_2 < 4)\) is also an example.