In many practical applications related to digital system design, it is a basic technique to use ternary-valued functions containing don't care. In this chapter, we discuss the methods for representing don't care by using BDDs. These methods can be extended to represent multi-valued logic functions which deal with integer values.

4.1 BOOLEAN FUNCTIONS WITH DON'T CARE

A Boolean function with don't care is regarded as a function from a Boolean vector input to a ternary-valued output, denoted as:

\[ f : \{0, 1\}^n \rightarrow \{0, 1, d\}. \]

where \( d \) means don't care. Such a function is also called an incompletely specified Boolean function. In the following sections, we simply call such a function ternary-valued function.

Ternary-valued functions are manipulated by the extended logic operations, and the rules of the logic operations between two ternary-valued functions are defined as follows:

<table>
<thead>
<tr>
<th>NOT</th>
<th>AND</th>
<th>OR</th>
<th>EXOR</th>
</tr>
</thead>
<tbody>
<tr>
<td>( f )</td>
<td>( \overline{f} )</td>
<td>( f \cdot g )</td>
<td>( f \lor g )</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>( d )</td>
<td>( d )</td>
<td>d</td>
<td>d</td>
</tr>
</tbody>
</table>

We also define two special unary operations \( f_r \) and \( f_l \) that are important since they are used for abstracting Boolean functions from ternary-valued functions:

\[
\begin{array}{c|c|c|c}
 f & [f] & [f] \\
 0 & 0 & 0 \\
 1 & 1 & 1 \\
 d & 1 & 0 \\
\end{array}
\]

In the operations of the ternary-valued functions, we sometimes refer to a constant function that always returns \( d \). We call it *chaos function*.

### 4.1.1 Ternary-Valued BDDs and BDD Pairs

There are two ways to represent ternary-valued functions by using BDDs. The first one is to introduce ternary values '0', '1' and 'd' at the terminal nodes of BDDs, as shown in Fig. 4.1. We call this BDD *ternary-valued BDD*. This method is natural and easy to understand, but it has a disadvantage that the operations \([f] \) and \([f] \) are not easy and that we have to develop a new BDD package for ternary-valued functions. Matsunaga et al. reported work[MF89] that uses such a BDD package.

The second way is to encode the ternary-value into a pair of Boolean values, and represent a ternary-valued function by a pair of Boolean functions, denote as:

\[ f : [f_0, f_1]. \]