



**Figure 6.6.** The BDDs (a)  $B_0$ , representing the constant 0 boolean function; similarly, the BDD  $B_1$  has only one node 1 and represents the constant 1 boolean function; and (b)  $B_x$ , representing the boolean variable  $x$ .

non-terminal node has exactly two edges from that node to others: one labelled 0 and one labelled 1 (we represent them as a dashed line and a solid line, respectively).

A BDD is said to be reduced if none of the optimisations C1–C3 can be applied (i.e. no more reductions are possible).

All the decision structures we have seen in this chapter (Figures 6.2–6.5) are BDDs, as are the constant functions  $B_0$  and  $B_1$ , and the function  $B_x$  from Figure 6.6. If  $B$  is a BDD where  $V = \{x_1, x_2, \dots, x_n\}$  is the set of labels of non-terminal nodes, then  $B$  determines a boolean function  $f(V)$  in the same way as binary decision trees (see Definition 6.3): given an assignment of 0s and 1s to the variables in  $V$ , we compute the value of  $f$  by starting with the unique initial node. If its variable has value 0, we follow the dashed line; otherwise we take the solid line. We continue for each node until we reach a terminal node. Since the BDD is finite by definition, we eventually reach a terminal node which is labelled with 0 or 1. That label is the result of  $f$  for that particular assignment of truth values.

The definition of a BDD does not prohibit that a boolean variable occur more than once on a path in the dag. For example, consider the BDD in Figure 6.7.

Such a representation is wasteful, however. The solid link from the left-most  $x$  to the 1-terminal is never taken, for example, because one can only get to that  $x$ -node when  $x$  has value 0.

Thanks to the reductions C1–C3, BDDs can often be quite compact representations of boolean functions. Let us consider how to check satisfiability and perform the boolean operations on functions represented as BDDs. A BDD represents a satisfiable function if a 1-terminal node is reachable from the root along a *consistent path* in a BDD which represents it. A consistent path is one which, for every variable, has only dashed lines or only solid lines leaving nodes labelled by that variable. (In other words, we cannot assign