BDD Evaluation with Restricted Variables

Prerequisite: mutually exclusive variables are grouped in the BDD variable order.

Let X = root variable of F, Y = root variable of G2. Then, if

(a) X, Y belong to different components:

$$\begin{aligned} &P(F) = \\ &= P(X)P(G1) + P(\overline{X})P(G2) \\ &= P(X)P(G1) + (1 - P(X))P(G2) \\ &= P(G2) + P(X)(P(G1) - P(G2)) \end{aligned} \quad \textbf{(1)}$$

(b) X, Y belong to the same component:

$$\begin{aligned} &P(F) = \\ &= P(ite(X, G1, G2)) \\ &= P(X \cdot G1 + \overline{X} \cdot G2) \\ &= P(X)P(G1) + P(\overline{X} \cdot (Y \cdot H1 + \overline{Y} \cdot H2)) \\ &= P(X)P(G1) + P(\overline{X}YH1 + \overline{X}\,\overline{Y}H2) \end{aligned} \quad \textbf{(2)}$$

Proof

X, Y belong to the same component $\Rightarrow Y = \overline{X}Y$ and $\overline{Y} = X + \overline{X}\,\overline{Y}$. Therefore we have

$$\begin{aligned} &P(G2) = \\ &= P(YH1 + \overline{Y}H2) \\ &= P(\overline{X}YH1 + (\overline{X}\,\overline{Y} + X)H2) \\ &= P(\overline{X}YH1 + \overline{X}\,\overline{Y}H2) + P(XH2) \\ &= P(\overline{X}YH1 + \overline{X}\,\overline{Y}H2) + P(X)P(H2)_{X=1} \\ &\Rightarrow P(\overline{X}YH1 + \overline{X}\,\overline{Y}H2) = P(G2) - P(X)P(H2)_{X=1} \end{aligned} \quad \textbf{(3)}$$

Substituting (3) into (2) gives

$$\begin{aligned} &P(F) = \\ &= P(X)P(G1) + P(G2) - P(X)P(H2)_{X=1} \\ &= P(G2) + P(X)(P(G1) - P(H2)_{X=1}) \end{aligned} \quad \textbf{(4)}$$

If the root variable of H2 does not belong to the same component as X, then $(H2)_{X=1} = H2$.

If the root variable of H2, eg, Z, does belong to the same component as X, then $(H2)_{X=1} = J2$ because when X equals one, Z must be zero.

In this case, (4) can be written as

$$P(F) = P(G2) + P(X)(P(G1) - P(J2)) \quad \textbf{(5)}$$

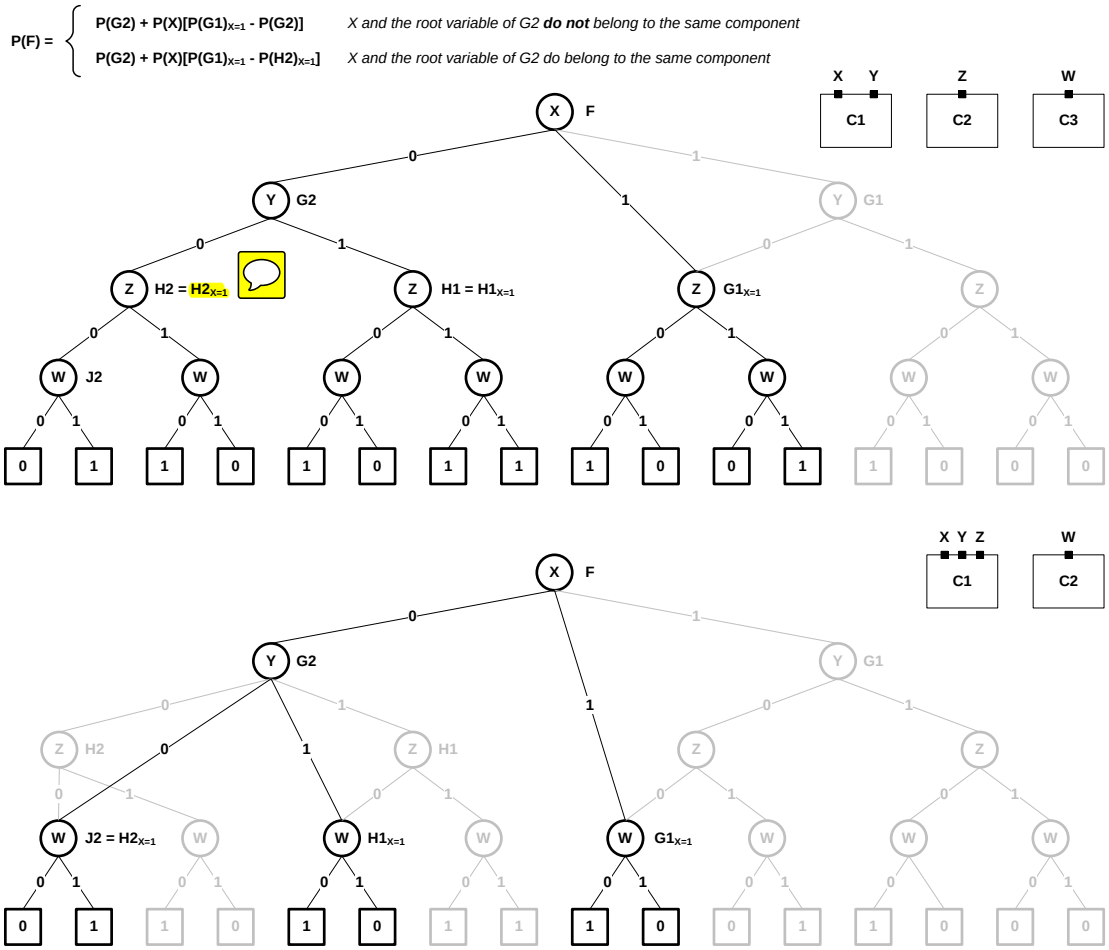

Figure 1a and 1b: Two examples for BDD evaluation with restricted variables (BDD ordered but not reduced for clarity)

(G1 – *true*-subgraph of F, G2 – *false*-subgraph of F, H1 – *true*-subgraph of G2, H2 – *false*-subgraph of G2, J2 – *false*-subgraph of H2; in Figure 1a, X and Y are s-dependent, in Figure 1b, X, Y and Z are.)

Details

The method depicted in Figure 1a and 1b is adapted from

> Zang, Wang, Sun, Trivedi (2003). A BDD-based algorithm for analysis of multistate systems with multistate components. *IEEE Transactions on Computers*, 52(12).

The effect of their MDO operations used in building the BDD can be likewise achieved by building the BDD in the usual way and then changing the way the *true*-subgraph of a node is traversed, if that node's variable is mutually exclusive with the root node or subsequent nodes of its *true*-subgraph (see Figure 1a and 1b). For the calculation of the probability of F we need the probability of the *true*-branch of the root (G1), and we know that, if the variable labelling the root node (for example, X) is *true*, all variables that are mutually exclusive (s-dependent) with X must be *false*. Thus, if the root node of G1 is s-dependent on X, instead of the "real" G1 we have to use a subgraph of G1, denominated $G1_{X=1}$, whose root variable is independent from X (if the root variable of G1 is independent already, then of course G1 is identical with $G1_{X=1}$). To find $G1_{X=1}$, starting from the root of G1 we follow the *false*-branch until encountering a node that is labelled with a variable independent from X. This is done recursively, resulting, for example, in a distinct $H1_{X=1}$ that is to G2 as $G1_{X=1}$ is to F and so on (Figure 1b). Note that H1 does not show up in the formula directly since it is just the G1 of the next recursion.

For the *false*-subgraph of a node the calculation method of Zang & al can be used as is: when the root node/variable X of the graph or a subgraph (F) has a variable Y as the root of its *false*-subgraph (G2) that is mutually exclusive with it, instead of G2 only, evaluation also uses the subgraph of Y that is reached by following the *false*-path rooted in Y to the next variable s-independent from X. That subgraph is denominated $H2_{X=1}$, because we use the H2 subgraph in a way, as if X were *true*: if the root variable of H2 is s-independent from X, we simply use H2 since the value of H2's variable is not affected by X being *true*; if the root variable of H2 is s-dependent on X (that is, mutually exclusive with X), its value has to be *false*, so $H2_{X=1}$, recursively, is replaced by the *false*-subgraph of H2 until $H2_{X=1}$ has a root variable that is s-independent from X (see Figure 1a and 1b).

To ensure correctness of the algorithm all variables that are s-dependent on each other must be adjacent in the variable order.