the algorithm again backtracks. It removes all assignments for B and sets $A = 1$. No forward implications are possible. PODEM backtraces from r, and goes through the intermediate objectives listed in Table 7.15. It sets $B = 0$. Forward implications yield $d = 0$, $X = 1$, $m = 1$, and $r = 1$. There is again a conflict at s because the fault is not sensitized, so PODEM again backtracks. It now sets $B = 1$, the alternate assignment for B. This yields the forward implications $d = 1$, $m = 1$, $r = 0$, $q = 0$, $s = \overline{D}$, $X = 1$, and $Y = \overline{D}$ so the fault is tested. Figure 7.24 shows the final circuit labeling where the test-pattern for the fault s sa1 is $ABC =$"11X" and the response $XYZ =$"0$\overline{D}X$."

If one repeats the above example with no controllability measures, where back-tracing always traces through the first logic gate input and first tries to achieve the objectives by setting that input to 0, then the exact same sequence of decisions will be followed. This does not mean that the controllability measure does not help, but instead indicates that this circuit example is pathological. In general, controllability measures greatly accelerate ATPG.

Algorithm 7.4 shows pseudo-code for the backtracing operation, Algorithm 7.5 shows objective setting, and Algorithm 7.6 shows PODEM pseudo-code.

Algorithm 7.4 *Backtrace* $(s,\ v_s)$*. Translate objective into PI assignment.*

```
{
   v = v_s;
   while (s is a gate output)
     {
       if (s is NAND or INVERTER or NOR) v = v̄;
       if (objective requires setting all inputs)
         Select unassigned input a of gate s with hardest controllability to value v;
       else
         Select unassigned input a of gate s with easiest controllability to value v;
       s = a;
     }
   /* s is now a primary input */
   return (s, v) /* Gate and value to be assigned */
}
```

Algorithm 7.5 *Objective* (g, v)

```
{ /*Target fault gate g stuck-at v */
    if (gate g is unassigned) return (g, v̄);
    Select a gate P from the D-frontier;
    Select an unassigned input l of P;
    if (gate P has controlling values) c = controlling input value of gate P;
    else if (0 value easier to get at input of XOR/EQUIVALENCE gate) c = 1;
```