

1 Part 3

In the last part, the existing implementation is to be extended by a practical application of BDD. Using BDDs, it is possible to symbolically represent a state-space. This representation allows to check quickly, whether a specific state is within the reachable state space or not. This part starts after the presentation of *Symbolic Traversal* in the lecture.

- Read the provided example below, it explains how BDDs are used to compute the reachable state space
- Download and integrate the source code with the interface for the extension from the OLAT platform
- Find the description of the methods in the interface
- Develop the *Reachability* class including test coverage (TDD optional). It inherits the functionality from your previously implemented Manager class and provides new functions to compute the reachable state space.
- The implementation is done, when our tests hold on the design. This time, also error handling should be considered as described in the function documentation.

2 Symbolic Traversal

The *Symbolic Traversal* algorithm is presented in the lecture in chapter 5. We now want to present an example of how the algorithm should be implemented within our BDD package. For a detailed description of the symbolic traversal, please refer to the lecture slides.

Given a state machine with two state variables s_0 and s_1 , no inputs and $S_{init} = (0, 0)$:

$$\delta = \begin{cases} s'_0 = \bar{s}_0 \\ s'_1 = \bar{s}_1 \end{cases}$$

1. Create variables for the current and next state s_0 , s_1 , s'_0 and s'_1

2. Compute the BDD for δ :

$$\delta_0 = \bar{s}_0, \delta_1 = \bar{s}_1$$

3. Compute the BDD for the transition relation $\tau = (s'_0\delta_0 + \bar{s}'_0\bar{\delta}_0) * (s'_1\delta_1 + \bar{s}'_1\bar{\delta}_1)$

4. Compute the BDD for the characteristic function of the initial state (0,0):

$$c_s = (s_0 == 0) * (s_1 == 0) = \overline{(s_0 \oplus 0)} * \overline{(s_1 \oplus 0)}$$

- 5.

$$c_{R_{it}} = c_s$$

- 6.

$$c_R = c_{R_{it}}$$

7. Compute the BDD for $img(s'_0, s'_1) = \exists_{s_0} \exists_{s_1} c_R * \tau$ by using the Manager functions:

$$temp1 = c_R * \tau$$

$$temp2 = coFactorTrue(temp1, s_1) + coFactorFalse(temp1, s_1)$$

$$img(s'_0, s'_1) = coFactorTrue(temp2, s_0) + coFactorFalse(temp2, s_0)$$

8. Compute $img(s_0, s_1) = \exists_{s'_0} \exists_{s'_1} (s_0 == s'_0) * (s_1 == s'_1) * img(s'_0, s'_1)$

$$temp1 = \overline{(s_0 \oplus s'_0)} * \overline{(s_1 \oplus s'_1)} * img(s'_0, s'_1)$$

$$temp2 = coFactorTrue(temp1, s'_1) + coFactorFalse(temp1, s'_1)$$

$$img(s_0, s_1) = coFactorTrue(temp2, s'_0) + coFactorFalse(temp2, s'_0)$$

9. Compute the BDD for the new $c_{R_{it}} = c_R + img(s_0, s_1)$

10. Check if $c_{R_{it}} == c_R$ In the first iteration, c_R consists of (0,0) and (1,1), whereas $c_{R_{it}}$ consists of (1,1). Therefore, it is not a fixed point and we have to go back to step 7 and perform a second iteration.

11. After the second iteration, $c_{R_{it}} == c_R$ holds and we reached a fixed point. c_R is now the symbolic representation of the set of reachable states. In this particular example, c_R represents the function $f = s_0s_1 + \bar{s}_0\bar{s}_1$ which is true if and only if the system is in a reachable state.