```
function REACH(bddfsm, init)
    reach ← init
    new ← POST(bddfsm, reach)
    while new ≠ INTERSECTION(reach, new) do
        reach ← UNION(DIFF(new, reach), reach)
        new ← POST(bddfsm, reach)
    end while
    return reach
end function

function BACKWARD_IMAGE_COMP(bddfsm, counter_examples, init)
    images ← []
    counter_example ← PICK_ONE_STATE_RANDOM(counter_examples)
    pre_counter_example ← counter_example
    counter_example_original ← counter_example
    APPEND(images, counter_example)
    while INTERSECTION(init, pre_counter_example) ≠ ∅ do
        counter_example ← pre_counter_example
        pre_counter_example ← PRE(bddfsm, counter_example)
        INSERT(images, pre_counter_example)
    end while
    return images, counter_example_original
end function

function FIND_TRACE(bddfsm, init, images, counter_example_original)
    trace ← []
    start ← init
    for i ← 1 to n do                                  ▷ LENGTH(images - 1)
        start ← INTERSECTION(start, images[i])
        next_state ← PICK_ONE_STATE(start)
        APPEND(trace, next_state)
        post ← INTERSECTION(POST(start), images[i + 1])
        inputs ← GET_INPUTS_BETWEEN_STATES(start, post)
        APPEND(trace, PICK_ONE_INPUTS(inputs))
        start ← post
    end for
    APPEND(trace, counter_example_original)
    return trace
end function
```

function CHECK_EXPLAIN_INV_SPEC((*bddfsm*, *spec*))
  $nspec \leftarrow \neg(spec)$
  $bddspec \leftarrow$ SPEC_TO_BDD($bddfsm, nspec$)
  $reach \leftarrow$ REACH($bddfsm, init$)
  if INTERSECTION($bddspec, reach$) $= \emptyset$ **then**
    return $(True, None)$
  else
    $counter\_examples \leftarrow$ INTERSECTION($bddspec, reach$)
    $images, counter\_example\_original \leftarrow$
      BACKWARD_IMAGE_COMP($bddfsm, counter\_examples, init$)
    $trace \leftarrow$
      FIND_TRACE($bddfsm, init, images, counter\_example\_original$)
    return $(False, trace)$
  end if
end function