```python
# reach.py

from pyeda.boolalg.bdd import (bddvar, expr2bdd, bdd2expr)
from pyeda.boolalg.expr import exprvar

def five_step_reach(rr, a, b, k):
    """
    Determine if two verticies a and b can reach eachother in five steps
    with the given BDD
    """

    # NOTE: Created this so you can vary the number of steps dynamically
    num_steps = 2
    var_base = 0
    var_coll = []

    # Get start expression covered
    var_set_start = [bddvar("v{}".format(i + var_base)) for i in range(k)]
    var_base += k
    var_coll.append(var_set_start)
    inter_set = set()

    # Now cover all intermediate
    for s in range(num_steps - 1):
        var_set = [bddvar("v{}".format(i + var_base)) for i in range(k)]
        var_base += k
        var_coll.append(var_set)
        for v in var_set: inter_set.add(v)

    # End expression
    var_set_end = [bddvar("v{}".format(i + var_base)) for i in range(k)]
    var_coll.append(var_set_end)

    # Variable-step reachability via composition
    hh = None
    past_var = None
    for curr_var in var_coll:

        if past_var is not None:

            # TODO: Fix problem with compose_dict
            compose_dict = {c:d for c, d in zip(past_var, curr_var)}
            new_comp = rr.compose(compose_dict)

            if hh is None: hh = new_comp
            else: hh &= new_comp

            # Remove quantifiers
            hh = hh.smoothing({sv for sv in curr_var})

        past_var = curr_var

    # Compute restrict dictionary
    restrict_dict = {c:d for c, d in zip(var_set_start, to_bin(a, k))}
    restrict_dict.update({c:d for c, d in zip(var_set_end, to_bin(b, k))})
    print("restrict_dict", restrict_dict)

    # See if node a can reach node b in given steps
    reachable = hh.restrict(restrict_dict)
    return reachable

def to_bin(num, k):
    node_bin = "{0:0b}".format(num)
    node_bin = "0" * (k - len(node_bin)) + node_bin
    return [int(x) for x in node_bin]
```