Formal Methods in SE: Homework #4

Due on 3/11 at 11:59 a.m.

Professor Ciardo

Charles Dudley

Problem 1

```
frbdd q_to_f(qrbdd x):
    // I am assuming I do not need to reduce terminals.
    //  I could have "return reduce(0)" but that seems unnecessary?
    if (f == 0):
        return 0
    if (f == 1):
        return 1

    // if f is redundant, redirect to a (reduced) child
    if (f[0] == f[1]):
        return q_to_f(f[0])

    // if f is not redundant, reduce each of its children
    h = new node at same level as f
    h[0] = q_to_f(f[0])
    h[1] = q_to_f(f[1])

    // using reduce() again to prevent introduction of duplicates
    return reduce(h)
```

Problem 2

```
integer cardinality(frbdd f):
    if (f == 0):
        return 0

    if (f[0] == 1):
        return 2^(f.lvl - 1) + cardinality(f[1])
    elif (f[1] == 1):
        return 2^(f.lvl - 1) + cardinality(f[0])
    else:
        return cardinality(f[0]) + cardinality(f[1])
```

Problem 3

```
// Let "accumulator" be a tuple of 3 fields:
//      bitstring:  array of bits (representing "path" to a node)
//      satisfying: boolean (representing if bitstring is satisfying)
//      explored:   a set of qrbdd nodes
bitstring low_sat_ass(qrbdd q):
    acc = (false, [], {})
    ret = low_sat_ass_rec(q, acc)
    if ret.satisfying:
        return ret.bitstring
    return None

accumulator low_sat_ass_rec(qrbdd q, accumulator acc):
    // add q to set of explored nodes
    acc.explored = Union(acc.explored, {q})

    // base case: bitstring has already been accumulated
    // simply return value of terminal as boolean
    if (q == 0):
        return acc
    if (q == 1):
        acc.satisfying = true
        return acc

    if (q[0] not in acc.explored):
        // check q[0] for satisfying assignment
        //  (an assignment from q[0] will always be "lower" than one from q[1])
        zero_acc = acc // make a copy of acc to pass recursively
        append(zero_acc.bitstring, 0) // append 0 to the bitstring
        zero_sol = low_sat_ass_rec(q[0], zero_acc)

        if zero_sol.satisfying:
            return zero_sol

    if (q[1] not in acc.explored):
        // check q[1] for satisfying assignment
        one_acc = acc
        append(one_acc.bitstring, 1)
        one_sol = low_sat_ass_rec(q[1], one_acc)

        if one_sol.satisfying:
            return one_sol

    // Can this ever happen in a qrbdd? I don't think so...
    // For both of the above "explored" checks to be false, q must
    //      be a duplicate node. There are no duplicates in qrbdds.
    return acc
```