COL215 Assignment 1 Report

Parth Patel (2021CS10550) and Raghav Ajmera (2021CS10562)

August 21, 2022

1 Notation

For 2x2, 4x2 and 4x4 K-maps, the REGION is a rectangle The coordinate of each element in this rectangle can be known by indices of the row and column in which it belongs to. *rows* and *columns* are two lists which contains the indices of row and column of all elements in the REGION.

So *rows* = [2,3] and *columns* = [2,3] describe a rectangle with top-left corner (2,2) and bottom-right corner (3,3). The lists *rows* and *columns* are indexed from 0 (i.e. *columns* = [0,1,2,3] when *term*[0] = *None* and *term*[1] = *None*

2 Implementation

We wrote separate codes for each possible length of TERM given in the input specification (i.e. 2, 3 and 4). For len(TERM) == 4, *columns* would contain 4 indices if first two elements of TERM are [*None*, *None*], 2 indices if first two elements of TERM are [*None*,1 or 0] or [1 or 0, *None*], 1 index if first two elements of TERM are [1 or 0, 1 or 0] Below is a code snippet which sets the *columns* list when len(TERM) == 4

```
    columns = []
    if len(term) == 4 :
        if term[0] == 1 :
            if term[1] == 0 : columns = [3]
            elif term[1] == 1 : columns = [2]
            else : columns = [2,3]

        elif term[0] == 0 :
            if term[1] == 0 : columns = [0]
            elif term[1] == 1 : columns = [1]
            else : columns = [0,1]

        else :
            if term[1] == 0 : columns = [0,3]
            elif term[1] == 1 : columns = [1,2]
            else : columns = [0,1,2,4]
```

By observing some repeated tasks that needed to be performed for each lengths (i.e. 2, 3, 4), we defined two functions, *execute*2 which is when rows or columns are 2 (used for 2x2 and 2x4 K maps) and *execute*4 when rows or columns are 4 (used for 2x4 and 4x4 K maps).

Input and Output specifications of both Helper functions - *execute*2 and *execute*4: INPUT - *term* , *k* ; OUTPUT - returns *rows* or *columns* according to the value of k (which is an index of *term*)

We reduced the lines of code required for defining the functions by observing some patterns (linear Lagrange interpolation)

```
def execute2(term,k) :
    l = []
    if term[k] != None : l = [term[k]]
    else : l = [0,1]
    return l

def execute4(term,k) : # returns rows or columns
    l = []
    if term[k] != None :
        if term[k+1] != None : l = [3*term[k]+((-1)**term[k])*term[k+1]]
        else : l = [2*term[k],2*term[k]+1]
    else :
        if term[k+1] != None : l = [term[k+1],3-term[k+1]]
        else : l = [0,1,2,3]
    return l
```

We further reduced the lines of code required for the program by observing more patterns and reduced the number of computations required, compared to first code.

To check if the REGION is legal or not, we checked if there exists an element 0 in the REGION. At the end, to ensure correctness of 'edge' cases, we exchange the elements in *columns* or *rows* if any of them is [0,3]. This is because the wrapped rectangles occur only when *rows* or *columns* is [0,3]

```
def execute4(term,k) : return ([3*term[k]+((-1)**term[k])*term[k+1]] if term[k+1] !=
    None else [2*term[k],2*term[k]+1]) if term[k] != None else ([term[k+1],3-term[k
    +1]] if term[k+1] != None else [0,1,2,3])

def execute2(term,k) : return [term[k]] if term[k] != None else [0,1]

def is_legal_region(kmap_function, term) :
    (columns, rows) = (execute4(term, 0), execute4(term, 2)) if len(term) == 4 else (
    execute4(term, 0), execute2(term, 2)) if len(term) == 3 else (execute2(term, 0),
    execute2(term, 1))

    isLEGAL = True  # to check if REGION is LEGAL
    for i in rows :
        for j in columns :
            if isLEGAL and kmap_function[i][j] == 0 : isLEGAL = False

    # to handle 'edge' cases
    if columns == [0,3] : columns = [3,0]
    if rows == [0,3] : rows = [3,0]

    return ((rows[0], columns[0]), (rows[-1], columns[-1]), isLEGAL)
```

3 Test Cases

```
>>>is_legal_region([[1,0,1,'x'],[1,1,0,'x'],[1,0,'x',1],[1,'x',0,1]],[None,0,None,0])
>>>((3, 3), (0, 0), True)
>>>is_legal_region([[1,'x'],[1,0]],[1, 0])
>>>((0, 1), (0, 1), True)
>>>is_legal_region([[0,1,1,0],['x',1,'x',0]],[1,0,None])
>>>((0, 3), (1, 3), False)
```