COL215 - Software Assignment 3

Deleting Unnecessary Terms

Deadline: 25 September 2022

1 Introduction

In this assignment, we will follow up on the Karnaugh Map region formation and expansion by attempting to delete unnecessary terms from our boolean expression, as the final stage in our strategy to minimise the literal count.

From the expanded function, delete those terms whose corresponding K-map regions are fully contained within one or more other terms/regions. Figure 1 below shows examples of terms that can be deleted because they are contained within other terms. Note that this containment of terms is defined for functions of any number of variables, even if we are not drawing the K-map.

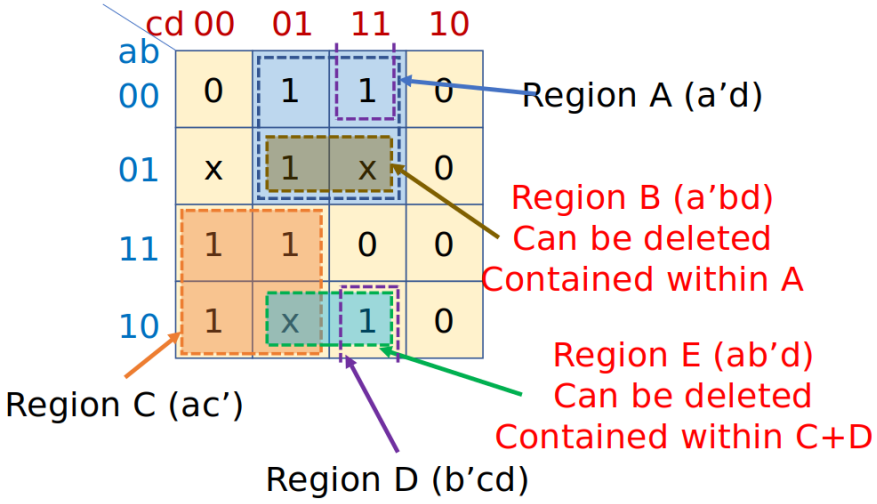


Figure 1: Reference example

2 Problem Description

Given a function with 0s, 1s, and x's for each input combination, print the minimised sum of product terms, where these terms are obtained by deleting as many as terms as possible from the expanded function of the input combination, so that the number of terms thus obtained, is minimised.

- Minimised terms set should cover all the cells with '1'. No such condition for cells with 'x'.
- You can use the K-map to display the given function and minimised terms (if the number of variables is small enough), but your implementation should work for a function with any number of input variables.
- The algorithm cannot take time that is exponential in the number of input variables (however, it is OK if the results are not optimal for all inputs).

Answer the following questions in your report:

- Explain your algorithm.
- Analyse the time complexity of your algorithm.
- Explain your testing strategy: what test cases did you run? Why is this set good enough to validate the implementation?

2.1 Graphic Module

You are given a utility (`K_map_gui_tk.py`) that takes as input a function's output values for all input combinations, and displays the corresponding K-map. Get familiar with the utility. Sample usage of the utility is provided in the `test.py` file. You will need to work with functions of 2 to 4 variables for the assignment.

2.1.1 Instructions to use the Module

1. creating a K-map (L is a n * m 2D list):

```
root = kmap(L)
```

2. drawing region with $(x1, x2)$ as the top-left coordinate, $(x2, y2)$ as the bottom-right coordinate:

```
root.draw_regions(x1, y1, x2, y2, color)
# color can be 'red', 'green', 'blue', 'yellow' etc
```

3. display the K-map:

```
root.mainloop()
```

2.1.2 Using the Code for Wrapping Region

```
from K_map_gui_tk import *

root = kmap([[0,1,1,0], ['x',1,'x',0], [1,0,0,0], [1,'x',0,0]])
root.draw_region(1,3,2,0,'blue')
root.draw_region(3,0,0,3,'green')
root.mainloop()
```

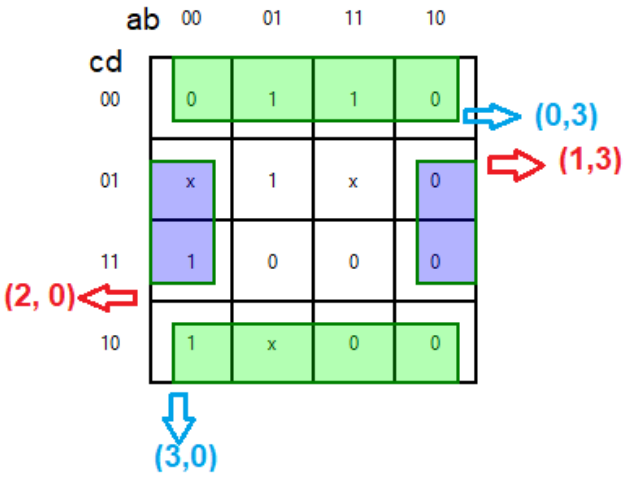


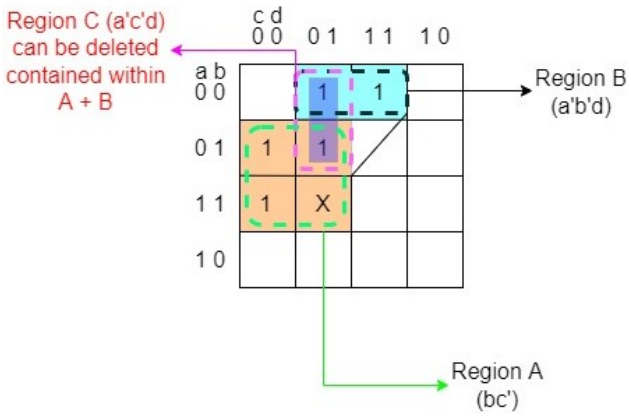
Figure 2: Output of the above code (arrows are for reference)

2.2 Test Cases

2.2.1 Sample Case I

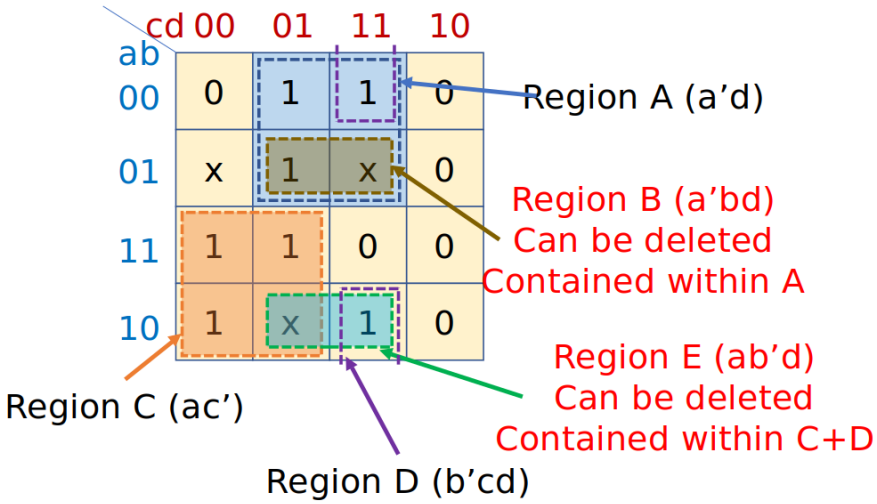
```
Input:
func_TRUE = ["a'bc'd'", "abc'd'", "a'b'c'd", "a'bc'd", "a'b'cd"]
func_DC = ["abc'd"]

Output:
["bc'", "a'b'd"]
```



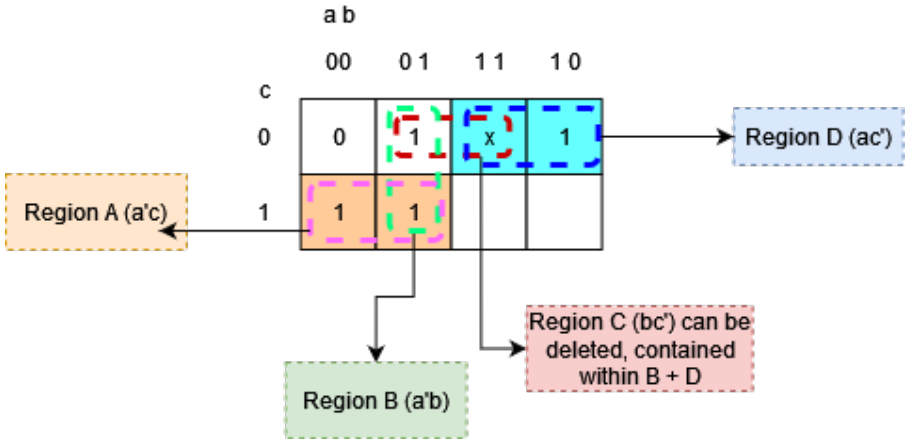
2.2.2 Sample Case II

```
Input:
func_TRUE = ["a'b'c'd'", "a'b'cd", "a'bc'd", "abc'd'",  "abc'd", "ab'c'd'", "ab'cd"]
func_DC = ["a'bc'd'", "a'bcd", "ab'c'd"]
Output:
["a'd", "ac'", "b'cd"]
```



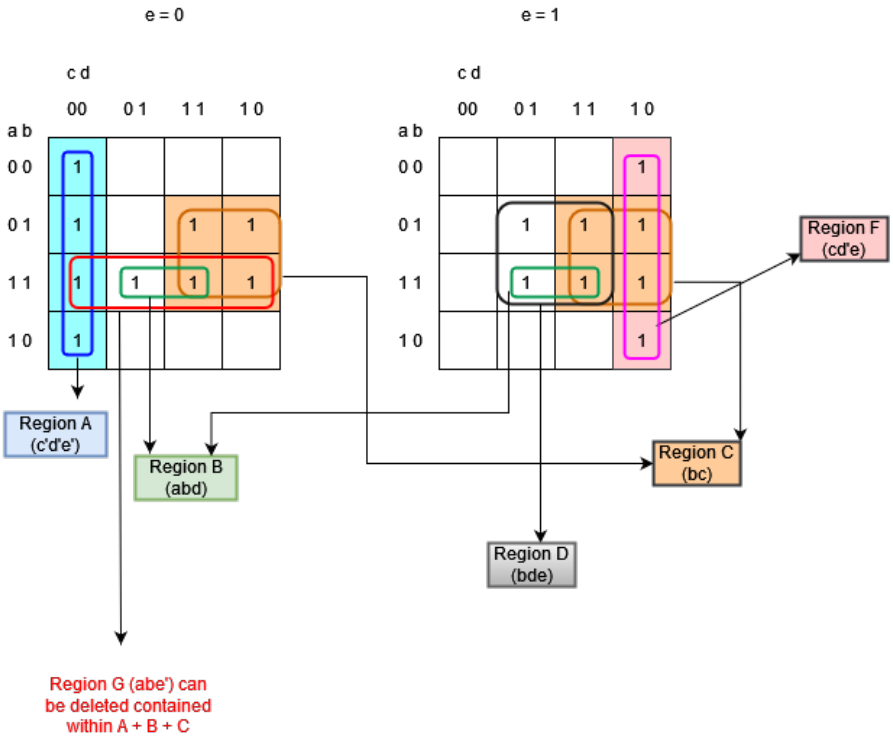
2.2.3 Sample Case III

```
Input:
func_TRUE = ["a'b'c", "a'bc", "a'bc'", "ab'c'"]
func_DC = ["abc'"]
Output:
["a'c", "a'b", "ac'"]
```



2.2.4 Sample Case IV

```
Input:
func_TRUE = ["a'b'c'd'e'", "a'bc'd'e'", "abc'd'e'", "ab'c'd'e'", "abc'de'", "abcde'",
             "a'bcde'", "a'bcd'e'", "abcd'e'", "a'bc'de", "abc'de", "abcde",
             "a'bcde", "a'bcd'e", "abcd'e", "a'b'cd'e", "ab'cd'e"]
func_DC = []
Output:
["c'd'e'", "abd", "bc", "bde", "cd'e"]
```



3 Submission Instructions

You are required to submit the following on Gradescope:

1. Python file named ⟨**entry_number1_number2**⟩**_assignment_3.py** containing the function

```
opt_function_reduce(func_TRUE, func_DC):
    """
        determines the minimum number of sum of product terms for the given K-map function
        Arguments:
            func_TRUE: list containing the terms for which the output is '1'
            func_DC: list containing the terms for which the output is 'x'
        Return:
            a list of minimum size containing terms:  terms in form of boolean literals
    """
    # your code here
```

2. Your report. No handwritten report, and document format should be pdf.
3. Work in your group of 2 students for the assignment. If you don't have a partner, it is OK; submit the assignment individually.
   - Groups will remain unchanged for subsequent assignments.
   - Only one submission per group is needed via gradescope. Gradescope help link: `https://help.gradescope.com/article/m5qz2xsnjy-student-add-group-members`
4. Post your doubts in the following thread on Piazza: `https://piazza.com/class/l6kqptxu7sz6i2/post/15`

3.1 Demo Instructions

In the demo to show the working of your implementation:

1. Print the terms which are in the region to be deleted.
2. The corresponding legal region, which already covers/contains the above terms.

Following examples are for illustration purpose, you can modify the way/language to print them. Here the terms are printed one by one, but you can print the terms belonging to same region in one line and in next line corresponding covering region.

```
N = 4

Term 1: "a'b'c'd"

Covering region: "a'b'd"

Term 2: "a'bc'd"

Covering region: "bc'"


N = 4

Term 1: "a'bc'd"

Covering region: "a'd"

Term 2: "a'b'cd"

Covering region: "a'd"

Term 3: "ab'cd"
```

```
Covering region: "b'cd"


N = 3

Term 1: "a'bc'"

Covering region: "a'b"

Term 2: "abc'"

Covering region: "ac'"

N = 5

Term 1: "a'b'c'd'"

Covering region: "c'd'e'"

Term 2: "a'b'c'd"

Covering region: "abd"

Term 3: "a'b'cd"

Covering region: "abd"

Term 4: "a'b'cd'"

Covering region: "bc"
```