COL215 SW2

Shreyansh Singh Aman Verma

October 28, 2022

1 Function Description

get_literals: This function takes in a term and returns the set of literals in it. For eg. input = ab'c, output = "a", "b' ", "c".

gray_code: This function returns the gray code for a given number.

get_coordinate: This function takes in a term and returns its location in kmap.

merge: This function takes in 2 terms and merges those terms if only one literal is different. For eg. ab'c' and ab'c will be merged to ab'.

end_points: This function takes in a kmap and a term and returns the start and end coordinates of the region representing that term.

draw_area: This function draws a region on the kmap gui.

comb_function_expansion: This function determines the maximum legal region for each term in the K-map function.

2 Approach

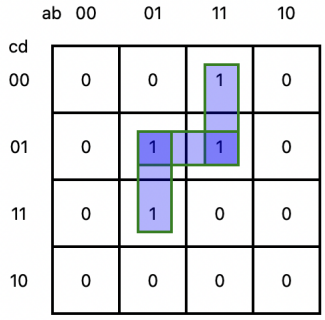
We first add all the terms with 1s and x's into a set. Then we look at each pair and merge them, if possible, and then add them into a new set. We repeatedly check the pairs in the new set, thus decreasing the number of literals in every new set. Thus we have a list of n+1 sets (where n is the number of variables). The sets in the end might be empty. The terms of the i_{th} ($i \in [0, n]$) set denotes legal region of size 2^i.

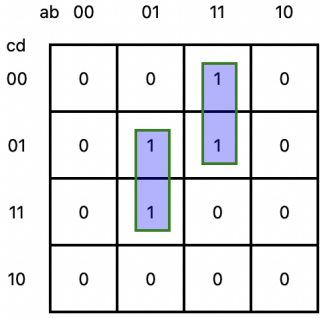
Now, for every term in the func_TRUE list, we want the term with the least number of literals (from the terms calculated earlier) which is a subset of this term. This essentially denotes the largest legal region the term is a part of. For this, we start our search from the n_{th} set (calculated earlier) till the 0_{th} set and we check all the terms of the set till we find the required term. On finding such a term the search ends and the term will denote the largest legal region.

Clearly a set of k terms will never produce more that k^2 terms in any of the subsequent sets.

Now, on assuming that the input consisted of n variables and t terms. Then in each iteration if the size of the set is s then O(s^2n) time is spent in this iteration. Also there are a total of n iterations. Clearly s is limited by t^2. Thus the overall time complexity of the algorithm is limited by $O(t^4n^2)$ which is polynomial in the input.

Ques1. Do all expansions result in an identical set of terms?
Ans: No not all expansions result in identical set of terms since the term may be expanded in different manners. For example observe the following:



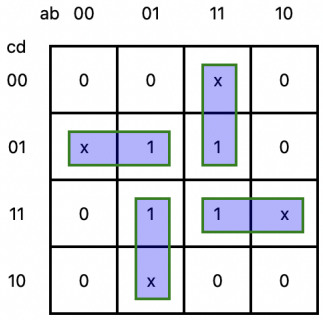


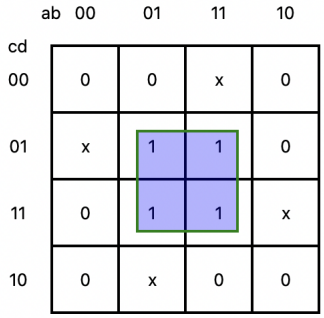
In this, the terms abc'd may be expanded to abc' or bc'd and so the expanded set of terms may not be identical.

Ques2. Are all expansions equally good, assuming that our objective is to maximally expand each term? Explain.
Ans: No, not all expansions are equally good. This is because pairing on one side might lead to a region of smaller size whereas pairing with some other element might lead to further grouping, thus forming larger groups. This is because a maximal expansion may just be a local maximum and not a global

maximum and expanding the term in a certain way doesn't lead to the largest region.

All expansions in the first figure are maximal but the expansion of the second figure seems more optimal.

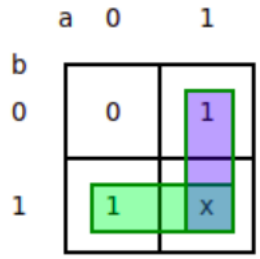




However, the manner in which we expand always leads to a maximum sized region - largest region and not just any maximal region.

3 Testing

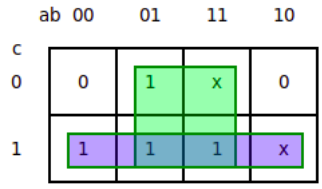
2 Variable :



2 variable Kmap

func_TRUE: [”a’b”, ”ab”’]
func_DC: [”ab”]

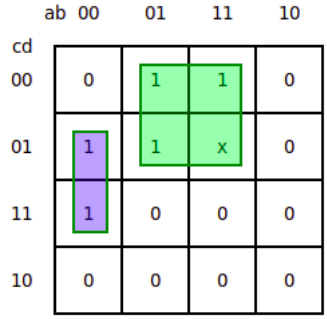
3 Variable :



3 variable Kmap

func_TRUE: [”a’bc”’, ”abc”, ”a’b’c”, ”a’bc”]
func_DC: [”abc”’, ”ab’c”]

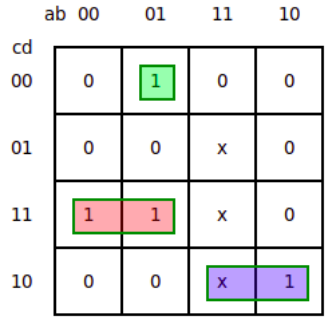
4 Variable :



4 variable Kmap

func_TRUE: [”a’bc’d”’, ”abc’d”’, ”a’b’c’d”, ”a’bc’d”, ”a’b’cd”]
func_DC: [”abc’d”]

4 Variable :



4 variable Kmap

func_TRUE: [”a’bc’d”’, ”ab’cd”’, ”a’b’cd”, ”a’bcd”]
func_DC: [”abc’d”, ”abcd”’, ”abcd”]

5 Variable :

```
func_TRUE: ["a'b'c'd'e'", "a'b'cd'e", "a'b'cde'", "a'bc'd'e'", "a'bc'd'e", "a'bc'de", "a'bc'de'", "ab'c'd'e'", "ab'cd'e'"]
func_DC: ["abc'd'e'", "abc'd'e", "abc'de", "abc'de'"]
ANS: ["c'd'e'", "a'b'cd'e", "a'b'cde'", "bc'", "bc'", "bc'", "bc'", "c'd'e'", "ab'd'e'"]
Legal Regions: {"c'd'e'", "a'b'cde'", "a'b'cd'e", "bc'", "ab'd'e'"}
```

5 variable Kmap

func_TRUE: [”a’b’c’d’e’”, ”a’b’cd’e”, ”a’b’cde’”, ”a’bc’d’e’”, ”a’bc’d’e”, ”a’bc’de”, ”a’bc’de’”, ”ab’c’d’e’”, ”ab’cd’e’”]
func_DC: [”abc’d’e’”, ”abc’d’e”, ”abc’de”, ”abc’de’”]

(** Kmap GUI does not support 5 variable kmap)

5 Variable :

```
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: []
ANS: ["c'd'e'", "c'd'e'", "c'd'e'", "c'd'e'", "abe'", 'bc', 'bc', 'bc', 'bc', 'bde', 'bde', 'bc', 'bc', 'bc', 'bc'
, "cd'e", "cd'e"]
Legal Regions: {'bc', "c'd'e'", "abe'", "cd'e", 'bde'}
```

5 variable Kmap

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: []