ASSIGNMENT-3

Deleting Unnecessary Terms

By-Priyadarshini(2021CS50606)

Hasita(2021CS50606)

ASSIGNMENT-3 REPORT

HELPERFUNCTIONS- regioncoordinates(var,list_region), intersection(x,y), subset(x,y), rc(n)

Algorithm and time complexity analysis -

Regioncoordinates is a helper function that converts the term in the expandedregion into coordinates representation of the region corresponding to the term. For example if the expandedregion contains an ad' term corresponding to 4 variable k-map its coordinate representation is [(3,0), (3,3),(4,0),(4,3)]. Regioncoordinates iterate over each term in expandedregion. It converts the term into a string with 1, 0, *s where 1 is allotted for the variable and 0 for its complement, and * is allotted if the corresponding variable is absent. The first for loop takes the term and converts it into string form in O(n) time. After getting the string form we are slicing it into its corresponding rows and column part and analyzing separately. We are iterating the rows' part and columns' part separately to get indices of columns and rows. ind1 and ind2 give the indices of rows and columns. Permutation of rows and column indices gives the corresponding region and all the permutations of rows and columns are obtained in O(m*n) time (where m is row indices length and n column indices length). Therefore, regioncoordinates take O(k*m*n) time because all the for loop are repeated k times (k= len(func_DC)). It returns a list of lists containing the region and the corresponding term.

Intersection is a helper function that takes two lists as inputs where each list is a region in the kmap and finds those cells in both rectangles which lie in the same column and returns the coordinates of those cells in the kmap and their corresponding indices in the input lists.

Subset is a helper function that returns True if x is a subset of y or else it returns False.

opt_function_reduce (func_TRUE, func_DC) takes coordinates from the regioncoordinates function and goes through for loops to give the final output. coordinates_only is a copy of coordinates containing only the coordinates. On iterating this through a for loop it finds whether a region lies in other regions or not. If a region is contained in one or more regions we append the regions in which it is contained, to a list and then finally append the region, which is

contained in all those regions, that is, the last element of the list is contained in the elements of the list before that. Then we are finding out the proper subsets, those which lie completely inside another region. Such regions can be deleted. Hence, we start by deleting those regions which completely lie inside an another region. Then we delete those regions which are contained in 2 or more other regions. Hence, this way we finally get the essential prime implicants and the required prime implicants and so as the total literal is decreased.

TIME COMPLEXITY:

The time complexity of our assignment 2 is O((m+n) ^2), where m is the length of func_TRUE and n is the length of func_DC.

The time complexity of opt_function_reduce(func_TRUE, func_DC) is O((m+n)^3) as there are three nested for loops in the function which iterate over the coordinate list.

Therefore the time complexity is O((m+n) ^3).

TESTCASES-

a) Testcases given in assignment:

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

func_DC = ["abc'd"]

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

2) 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", "bc'", "ac'"]

3) Input: func_TRUE = ["a'b'c", "a'bc", "a'bc'", "ab'c'"]

func_DC = ["abc'"]

Output: ["a'c", "abc'", "ab'c'"]

b) Additional testcases for verification-

1) Input: func_TRUE=[“a’b’c’d’”,” a’b’c’d”,”a’bc’d’”,”a’bc’d”,”a’bc’d”,”abc’d’”,”abc’d”,”ab’c’d’”,”ab’c’d”]

func_DC = [“abcd’”,” a’b’cd’”]

Output: ["c'd", "bc'", "ac'", "acd'"]

2) Input:
func_TRUE=["a'b'c'd","a'b'cd","a'bc'd'","a'bc'd","abc'd'","abc'd","ab'c'd'"
,"ab'c'd","ab'cd"]

func_DC = ["a'bcd","ab'cd'"]

Output: ["a'd", "bc'", "ab'"]

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