COL 215 - SW Assignment 3

Manshi Sagar 2020CS50429
Richa Yadav 2020CS50438

November 2022

1 Explain your algorithm

- First, we converted the TRUE and DON'T-CARE terms into binary form and then stored them in a column (sorted according to the number of '1' in terms).
- Grouped the terms according to number of 1's and then in two consecutive groups(no. of 1's differ by only 1), we are checking if there are 2 terms one in each such that they differ in only one position, if yes, then are we are adding these terms with substituting that index with '-', to modified list that will be used in next iteration for grouping and further reduction.
- When there is no further reduction possible then these terms are added to prime implicants list, then using this list we are finding out the minterms with respect to every PI that are covered by PI. After this, we are making dictionary whose keys are minterms and values are the reduced terms that cover those minterms. Then we are checking if there is any key that has only one value then this term is essential prime implicant.
- Then there is no minterm that is covered by only one implicant so, we'll have to fine minimised set of terms that cover all these minterms.
- Combining both the lists for final result.

2 Analyse the time complexity of your algorithm

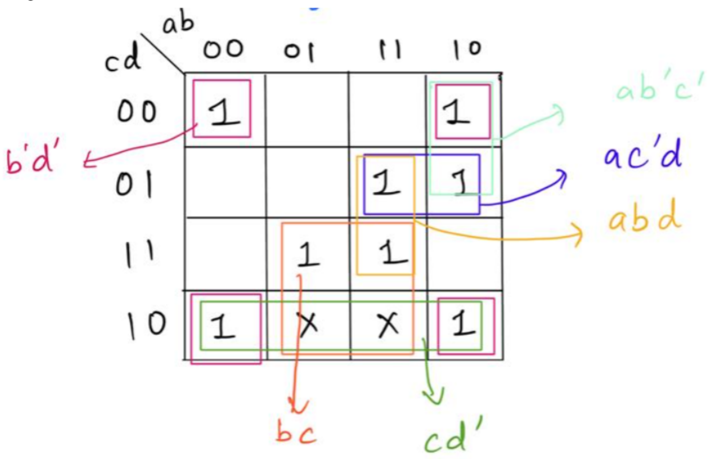
t = total number of True and Don't Care terms
n = number of variables
Process of combining minterms can take place atmost n times, taking O(t^2) time in each iteration.
Time of this step = O(n.t^2)
Time taken to make matrix to store initial terms and their expanded minterm = O(t.n)
Time taken to find essential prime minterms = O(t)
Time taken to find terms for left minterms that are not covered by single implicant = O(t^3)
.
Hence, total time complexity of algorithm = O(n.t^2)

3 Explain your testing strategy

We tried making testcases in which max-legal-regions of some TRUE terms could be contained in max-legal-regions of other minimised terms.

- T = ["a'b'c'd'", "ab'c'd'", "abc'd", "ab'c'd", "a'bcd" , "abcd" , "a'b'cd'", "ab'cd'"]
  DC = ["a'bcd'", "abcd'"]
  cd' is covered by ["b'd'"]
  abd is covered by ['bc', "ac'd"]
  ab'c' is covered by ["b'd'", "ac'd"]

Final reduced minterms : [’bc’, ”b’d’”, ”ac’d”]



- T = [”abc’d’”, ”a’b’c’d”, ”ab’c’d”, ”a’b’cd”, ”a’bcd’”]
  DC = [”a’bc’d’”, ”a’bc’d”, ”a’bcd”, ”ab’cd”, ”abcd’”]
  Result = [”c’d”, ”a’db”]

- T = [”a’b’c’d’”, ”a’bc’d’”, ”abc’d’”, ”a’bcd’”, ”abcd’”]
  DC = [”abc’d”, ”abcd”, ”a’b’cd’”]
  ab is covered by [”bd’”]
  Final reduced minterms : [”a’d’”, ”bd’”]

- T = [”a’b’c’d’e’”, ”a’bcd’e’”, ”abcd’e’”, ”a’b’cd’e”, ”a’bcd’e”, ”abc’d’e”, ”a’b’cde”, ”abcde’”, ”ab’c’de’”]
  DC = [”a’bc’d’e’”, ”abc’d’e’”, ”ab’c’d’e’”, ”a’bc’d’e”, ”abcd’e”, ”a’b’c’de’”, ”abc’de’”]
  a’cd’e is covered by [”bd’”, ”a’b’ce”]
  ac’e’ is covered by [”b’c’e’”]
  c’d’e’ is covered by [”b’c’e’”]
  Final reduced minterms : [”bd’”, ”abe’”, ”a’b’ce”, ”b’c’e’”]

- T = [”a’b’c’d’e’”, ”a’b’c’d’e”, ”a’bc’d’e”, ”ab’cd’e”, ”a’b’cde”, ”abcde”, ”a’bcde’”, ”abcde’”]
  DC = [”ab’c’d’e’”, ”a’b’cd’e”, ”abc’d’e”, ”ab’c’d’e”, ”a’bcde”, ”ab’cde”, ”a’b’cde’”, ”ab’cde’”]
  b’d’e is covered by [”c’d’e”, ”b’c’d’”, ”b’ce”]
  Final reduced minterms : [’cd’, ”c’d’e”, ”b’c’d’”, ”b’ce”]

- T = [”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”]
  DC = []
  abe’ is covered by [’abd’, ’bc’, ”c’d’e’”]
  bd’e’ is covered by [’bc’, ”c’d’e’”]
  Final reduced minterms : [’bde’, ’bc’, ”cd’e”, ”c’d’e’”, ’abd’]