Report

COL215P Assignment - SW2

Somaditya Singh | 2020CS10389

Ujjwal Mehta | 2020CS10401

Overview

The function `comb_function_expansion` takes the list of terms that correspond to `true` and `don't care` values of the function, and for each term, returns the maximally reduced term.

Design Logic

We have used the following helper functions:

- `break_literal` : converts the string corresponding to a term to a list of characters.
- `produced_terms` : takes a list of characters corresponding to the term and returns a list of terms (in list form) consisting of all possible binary values of the terms, i.e., if we input ["a", "b"], it returns "a", "b"], ["a'", "b"], ["a", "b'"], ["a'", "b'".
- `all_possible_terms_check` : takes as input the literals that have already been combined and the literals that are remaining and checks if all possible combinations of the remaining term with the combined literals are present.
- `maximal_expansion` : takes as input the dictionary of terms that are `TRUE` or `NOT CARE`, and the term to be reduced, and reduces the first literal possible.
- `comb_function_exapnsion` : calls `maximum_expansion` on each term.

Time Complexity

Since the maximum number of checks to reduce a literal in the given term can't exceed the size of the input list, the time complexity is polynomial in the size of the input list.

Do all expansions result in identical set of terms?

No. Consider the set ["a'bc", "abc", "abc']. Then the term `abc` can be maximally reduced to both `bc` or `ab` by combining with different terms.

Are all expansions equally good?

Consider the set ["abc", "a'bc", "ab'c'", "abc'", "ab'c"] then the term `abc` can be maximally reduced to `bc` if we combine with `a'bc` but it can also be reduced to `a` if we use the other terms.

Testcases

Case 1:

Input:

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

Output:

```
["bc'", "bc'", "a'c'd", "bc'","a'b'd"]
```

Case 2:

Input:

```
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'"]
```

Output:

```
["c'd'e'", "a'b'cd'e", "a'b'cde'", "c'd'e'", "bc'", "bc'", "bc'",
"c'd'e'", "ab'd'e'"]
```

Case 3 :

Input :

```
func_TRUE = ['abc',"a'bc"]
func_DC = ["abc'","ab'c","ab'c'"]
```

Output:

```
['bc', 'bc']
```