Technical Manual for Tabulation Method – Quine McCluskey

By: Dorado, Abram & Malimban, Kristine Jewel

Global Variable Code: - (global's.py)

==========

def initialize(): <- Function

Description: The variables in this function is declared global meaning all the different python files can access and will have the same value of this variable.

Variables:

var4	Stores the print_primary() function to this variable
vari4	Stores 3 variables coming from printing the table for the minterms
chart	Stores the prime_charts() function
txt	Stores the printing of All Prime Implicants and its values
unmarked1	Stores the converted tuple of unmarked elements of prime implicants
space	Stores the newline for spacing in printing
status	Stores the label of status of table
error	Stores the warning sign for wrong input

-These variables are made for the GUI so that it will be accessed and used.

Python Functionality Code: - (print.py)

==========

def searchEPI(primeChart): <- Function

Description: finds essential prime implicants for it to be displayed

Arguments:

primeChart	prime implicants chart

Returns:

res	a refined list

---

def getVariables(iteration, variable): <- Function

Description: find variables in a minterm (For example, the minterm --01 has C' and D as variables)

Arguments:

iteration, variable	number from an iteration. variable/letters to be used for the final output

Returns:

Var_list	variable list

---

def multiplyMinterms(num1,num2): <- Function

Description: multiplies minterms and appends them on the list

Arguments:

num1,num2	the numbers from interaction

Returns:

res	a refined list

---

def multiplyExpressions(num1,num1): <- Function
Description: multiplies expressions and appends them on the list
Arguments:
num1 , num1 the numbers from interaction
Returns:
res a refined list

---

def refineList(givenList,dontCareList): <- Function
Description: removes the don't care terms from the list
Arguments:
givenList, dontCareList list to be edited, dont care list
Returns:
res a refined list

---

def flattenList(groupCopy): <- Function
Description: This function removes the multidimensions in a list (flattens the list)
Arguments:
groupCopy accepts a copy of a group and flattens the list
Returns:
flattened_items returns the a flattened list

---

def getMergeMinterms(iteration): <- Function
Description: This function finds which minterms are merged. For example, we can get -001 is obtained by merging 9(1001) and 1(0001)
Arguments:
iteration Accepts the number from an iteration
Returns:
temp a temporary list

---

def compareMinterms(minterm1,minterm2): <- Function
Description: function for checking if 2 minterms differ by 1 bit only
Arguments:
minterm1,minterm2 minterms that will be compared
Returns:
True, mismatch_index boolean value (true or false) if the minterms only differ by 1 bit

---

def removeTerms(chart,terms): <- Function
Description: removes minterms which are already covered from chart
Arguments:
chart, terms chart, terms to check
Returns:
None

---

def grouping(minterms, groups, size): <- Function
Description: groups the primary minterms
Arguments:
minterms, groups, size the entered minterms plus dontCC, groupings, get size
Returns:
None
---
def print_primary(groups): <- Function
Description: print the primary groups
Arguments:
groups groupings of minterms
Returns:
None
---
def convertTuple(tup): <- Function
Description: This is an auxiliary Function that converts a tuple into a string
Arguments:
tup The tuple that will be converted
Returns:
str Returns the prime implicants and the primary answer
---
def prime_charts(enter_minterms, all_prime, enter_dontCC, variable): <- Function
Description: last function for printing and processing of Prime Implicant chart
Arguments:
enter_minterms, all_prime, enter_dontCC, variable The minterms that was inputted, the don't care conditions, and all the prime implicants calculated. The variable that will be passed for the final output
Returns:
str1 + str2 + str3 +str4 Returns ina concatenated style the printed output of the tables
---
def driver(variable,minterms, dontCare, sizing): <- Function
Description: The main function is the one responsible for setting up the tabular method program., initializing the minterms and calling the functions within the program
Arguments:
variable, minterms, dontCare, sizing The variable that will be passed for the final output. The entered minterms plus dontCC, groupings, get size

Returns:
None

Kivy (GUI) Code: - (screen.py)

==========

screen_helper - This code will generate all the commands regarding the GUI, things such as font size, font color, and location of elements in the window. Inside the screen helper are the contents below.

ScreenManager: **-** This screen manager gives the overall layout of the screen switching in the GUI, the contents of this screen manager are as follows:

-Menu
-Input_User
-Results
-Results2
-Results3

<Menu>: **-** This menu is the first page of the program where the user can see the intro of the Program. In here we will implement and modify some contents to be seen in the GUI

Widget: -The widget is the overall canvas of the GUI window
canvas: -In canvas is where we set the background color and the rectangle size of the slide
FloatLayout: **-**This means that you can freely customize the layout of the GUI, as you desire.
Image: -declaring this code, you will be able to insert an image in the GUI
Label: -This line of code will produce a label in the GUI that can be manipulated.
Button: -You can make a button and adjust it as you desire

<Input_User>: **-**This screen is for the user input, where the minterms will be typed.
MDTextField: -This is to create a text field for the user input.

<Results>: -The screen is for the first results of the table.
<Results2>: **-** The screen is for the 2nd results of the table.
<Results3>: **-** The screen is for the final results of the table.

LabelBase.register -used for registering the font in the Kivy

class Input_User(Screen): - used of class for screen
def __init__(self, **kwargs): <- Function
Description: The function for the Kivy to generate the screens
Arguments:
self, **kwargs Variable for the program to run

Returns:
None

---

def inputs(self, variable, minterms, dontCare, sizing): <- Function
Description: The function of the 3rd to the last screen
Arguments:

Self, variable, minterms, dontCare, sizing	Variable for the program to run. The variable that will be passed for the final output. The entered minterms, plus dontCC, groupings, get size

Returns:

ntext	this returns the printing of input of minterms and variable

---

def input1(self, variable, minterms, dontCare, sizing): <- Function
Description: The function of the second screen
Arguments:

Self, variable, minterms, dontCare, sizing	Variable for the program to run. The variable that will be passed for the final output. The entered minterms, plus dontCC, groupings, get size

Returns:

ntext	this returns the printing of Unmarked element label and printed table

---

def input2(self, variable, minterms, dontCare, sizing): <- Function
Description: The function of the 2nd to the last screen
Arguments:

Self, variable, minterms, dontCare, sizing	Variable for the program to run. The variable that will be passed for the final output. The entered minterms, plus dontCC, groupings, get size

Returns:

globals.status + globals.unmarked1 + globals.space + variable2 + ntext + globals.error	returns the printing of the status o table, the Unmarked elements, the spaces, the printed table and the error catcher

---

class TabulationMethod(MDApp):
def build(self): <- Function
Description: The main function that is responsible of running the Kivy code
Arguments:

MDApp	kivy import for building the screen

Returns:

Screen	It returns the screen output