Biological Computation – Final Project

Yaniv Hajaj

Avidan Menashe

https://github.com/YanivHajaj/Bio-Computation-Final-Project/tree/master

1. Write a computer program (in your favourite programming language) that finds all the monotonic regulation conditions of the reasoning engine, as we studied in class and appear in the following table (focus only on part d which is the main table). The program should show that the 18 rows correspond to the only regulation conditions that satisfy monotonic requirement and consider whether none, some or all of the activators / inhibitors are present. Please write a short explanation of how your program works, a readme on how to run the program and a printout of the output. Please also open a github repository where all files are made available.

יש 4 משפיעים על הגן האמצעי (שני אקטיבטורים ושני אינהיביטורים) נסמן אותם ברבעייה הסדורה (act1, act2, inh1, inh2):

```
#(act1, act2, inh1, inh2)
unique_scenarios = [
    (0, 0, 0, 0),
    (1, 0, 0, 0),
    (1, 1, 0, 0),
    (0, 0, 1, 0),
    (1, 0, 1, 0),
    (1, 1, 1, 0),
    (0, 0, 1, 1),
    (1, 0, 1, 1),
    (1, 1, 1, 1),
]
```

יש תרחישים דומים כמו למשל (0, 0, 1, 0) הוא תרחיש דומה ל (1, 0, 0, 0) כי בשניהם יש שני אקטיבטורים כבויים ואינהיביטור 1 דלוק לכן בסהכ יהיו 9 תרחישים שונים ולא $2^4 = 16$.

ראשית נשתמש בספריה :

```
from itertools import product
```

כדי ליצור כל קומבינציה של פונקציה

```
# Generate all possible Boolean functions for these scenarios
all_functions = list(product([0, 1], repeat=len(unique_scenarios)))
```

כעת נשים לב שאם אנחנו בעמודה 3 הפונ היא 1 לוגי לכן על פונ שלא מקיימת זאת נפסלת, כנל לגבי פונ שבעמודה השישית שלה אין 0, היא נפסלת.

```
def is_monotonic(function):
    # Ensure (1, 1, 0, 0) is always 1
    if function[2] != 1:
        return False

    # Ensure (0, 0, 1, 1) is always 0
    if function[6] != 0:
        return False
```

כעת נעבור על כל שני עמודות עמודה (כל עמודה מייצגת תרחיש, למשל 0,0,0,0 כולם כבויים שקול לעמודה ראשונה בהרצאה) ונבדוק את יחס השקילות קטן שווה ל, שהוא בעצם יהווה לנו את תכונת המונוטוניות שחייבת להתקיים עבור כל פונקציה .
למשל שלושת העמודת הראשונות מההרצאה מקיימות:

(0, 0, 0, 0)<=(1, 0, 0, 0)<=(1, 1, 0, 0)=1

מאחר ואם אנחנו מכבים אקטיבטור אזי ערך הפונ חייב להישאר זהה או לרדת.

```
    for i in range(len(function)):
        for j in range(len(function)):
            if is_less_or_equal(unique_scenarios[i], unique_scenarios[j]) and function[i] > function[j]:
                return False
    return True

```

נשים לב שאנחנו בודקים אם מתקיים יחס השקילות בעמודה i קטן שווה לעמודה j ובנוסף לכך function[i] גדול מfunction[j] אזי נוכל לקבוע שהפונ פסולה כי היא אינה מקיימת מונוטוניות.

את ערך השקילות קטן שווה בדקנו בדרך הבאה:
עבור שני עמודות שונות, אם שני אקטיבטורים שווים אז נבדוק את ההינהבטורים, במידה והם מקיימום מונוטוניות נחזיר TRUE).
באופן דומה עבור שני עמודות שונות, אם שני הינהבטורים שווים אז נבדוק את אקטיבטורים, במידה והם מקיימום מונוטוניות נחזיר TRUE).

```
#check if scenario1 is less then or equal scenario2
def is_less_or_equal(scenario1, scenario2):
    #if all inhibitors the same check
    if(scenario1[2]==scenario2[2] and scenario1[3]==scenario2[3]):
        if (scenario1[0]<=scenario2[0] and scenario1[1]<=scenario2[1]):
            return True

    #if all activators the same check
    if(scenario1[0]==scenario2[0] and scenario1[1]==scenario2[1]):
        if(scenario1[2]>=scenario2[2] and scenario1[3]>=scenario2[3]):
            return True
    return False
```

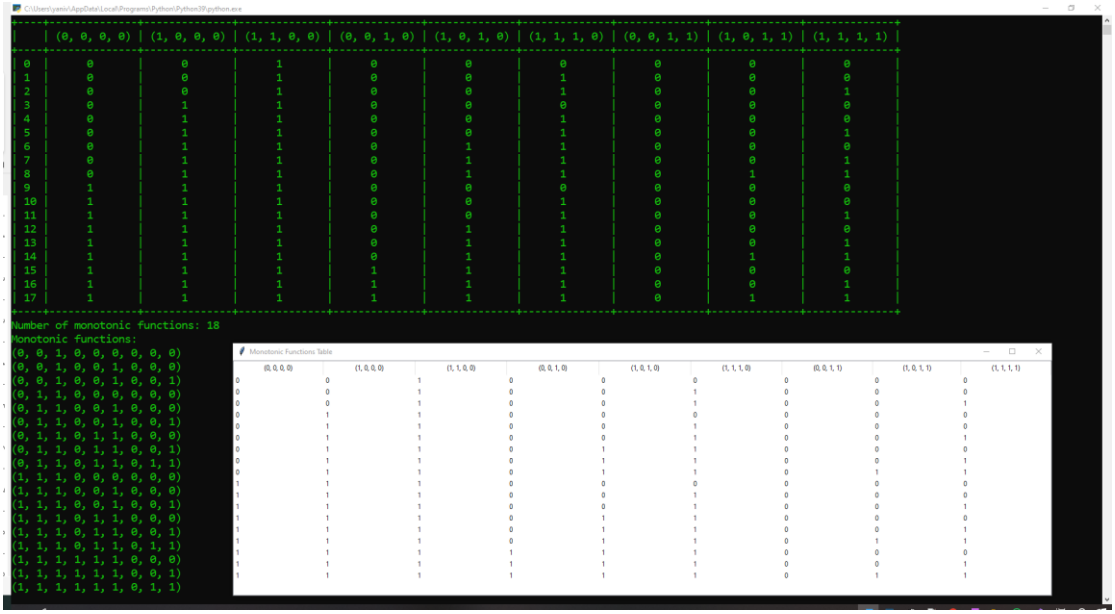
לבסוף נדפיס את התוצאות בקונסול וגם בטבלה חיצונית :

```
# Print the console output
print(tabulate(df, headers='keys', tablefmt='pretty'))
print(f"Number of monotonic functions: {len(monotonic_functions)}")
print("Monotonic functions:")
for function in monotonic_functions:
    print(function)

# Show the table in a new window
show_table()

########################################################################## print
```

ונקבל את התוצאות:



	(0, 0, 0, 0)	(1, 0, 0, 0)	(1, 1, 0, 0)	(0, 0, 1, 0)	(1, 0, 1, 0)	(1, 1, 1, 0)	(0, 0, 1, 1)	(1, 0, 1, 1)	(1, 1, 1, 1)
0	0	0	1	0	0	0	0	0	0
1	0	0	1	0	0	1	0	0	0
2	0	0	1	0	0	1	0	0	1
3	0	1	1	0	0	0	0	0	0
4	0	1	1	0	0	1	0	0	0
5	0	1	1	0	0	1	0	0	1
6	0	1	1	0	1	1	0	0	0
7	0	1	1	0	1	1	0	0	1
8	0	1	1	0	1	1	0	1	1
9	1	1	1	0	0	0	0	0	0
10	1	1	1	0	0	1	0	0	0
11	1	1	1	0	0	1	0	0	1
12	1	1	1	0	1	1	0	0	0
13	1	1	1	0	1	1	0	0	1
14	1	1	1	0	1	1	0	1	1
15	1	1	1	1	1	1	0	0	0
16	1	1	1	1	1	1	0	0	1
17	1	1	1	1	1	1	0	1	1

Number of monotonic functions: 18
Monotonic functions:
(0, 0, 1, 0, 0, 0, 0, 0, 0)
(0, 0, 1, 0, 0, 1, 0, 0, 0)
(0, 0, 1, 0, 0, 1, 0, 0, 1)
(0, 1, 1, 0, 0, 0, 0, 0, 0)
(0, 1, 1, 0, 0, 1, 0, 0, 0)
(0, 1, 1, 0, 0, 1, 0, 0, 1)
(0, 1, 1, 0, 1, 1, 0, 0, 0)
(0, 1, 1, 0, 1, 1, 0, 0, 1)
(0, 1, 1, 0, 1, 1, 0, 1, 1)
(1, 1, 1, 0, 0, 0, 0, 0, 0)
(1, 1, 1, 0, 0, 1, 0, 0, 0)
(1, 1, 1, 0, 0, 1, 0, 0, 1)
(1, 1, 1, 0, 1, 1, 0, 0, 0)
(1, 1, 1, 0, 1, 1, 0, 0, 1)
(1, 1, 1, 0, 1, 1, 0, 1, 1)
(1, 1, 1, 1, 1, 1, 0, 0, 0)
(1, 1, 1, 1, 1, 1, 0, 0, 1)
(1, 1, 1, 1, 1, 1, 0, 1, 1)

Monotonic Functions Table

(0, 0, 0, 0)	(1, 0, 0, 0)	(1, 1, 0, 0)	(0, 0, 1, 0)	(1, 0, 1, 0)	(1, 1, 1, 0)	(0, 0, 1, 1)	(1, 0, 1, 1)	(1, 1, 1, 1)
0	0	1	0	0	0	0	0	0
0	0	1	0	0	1	0	0	0
0	0	1	0	0	1	0	0	1
0	1	1	0	0	0	0	0	0
0	1	1	0	0	1	0	0	0
0	1	1	0	0	1	0	0	1
0	1	1	0	1	1	0	0	0
0	1	1	0	1	1	0	0	1
0	1	1	0	1	1	0	1	1
1	1	1	0	0	0	0	0	0
1	1	1	0	0	1	0	0	0
1	1	1	0	0	1	0	0	1
1	1	1	0	1	1	0	0	0
1	1	1	0	1	1	0	0	1
1	1	1	0	1	1	0	1	1
1	1	1	1	1	1	0	0	0
1	1	1	1	1	1	0	0	1
1	1	1	1	1	1	0	1	1

ניתן לראות שלאחר כל האילוצים אנחנו מסיימים עם 18 פונקציות שונות בדומה למה שראינו בהרצאה: