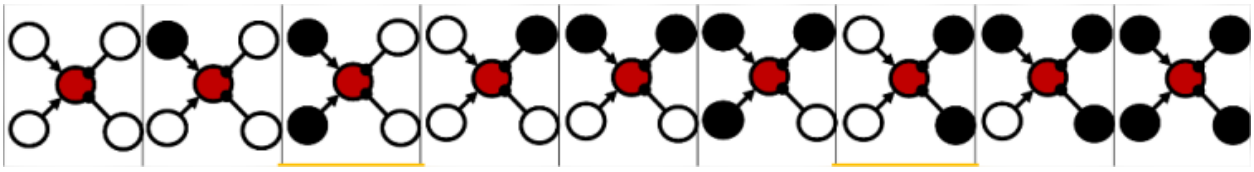
# פרויקט גמר בקורס חישוב ביולוגי


סמי נחמד

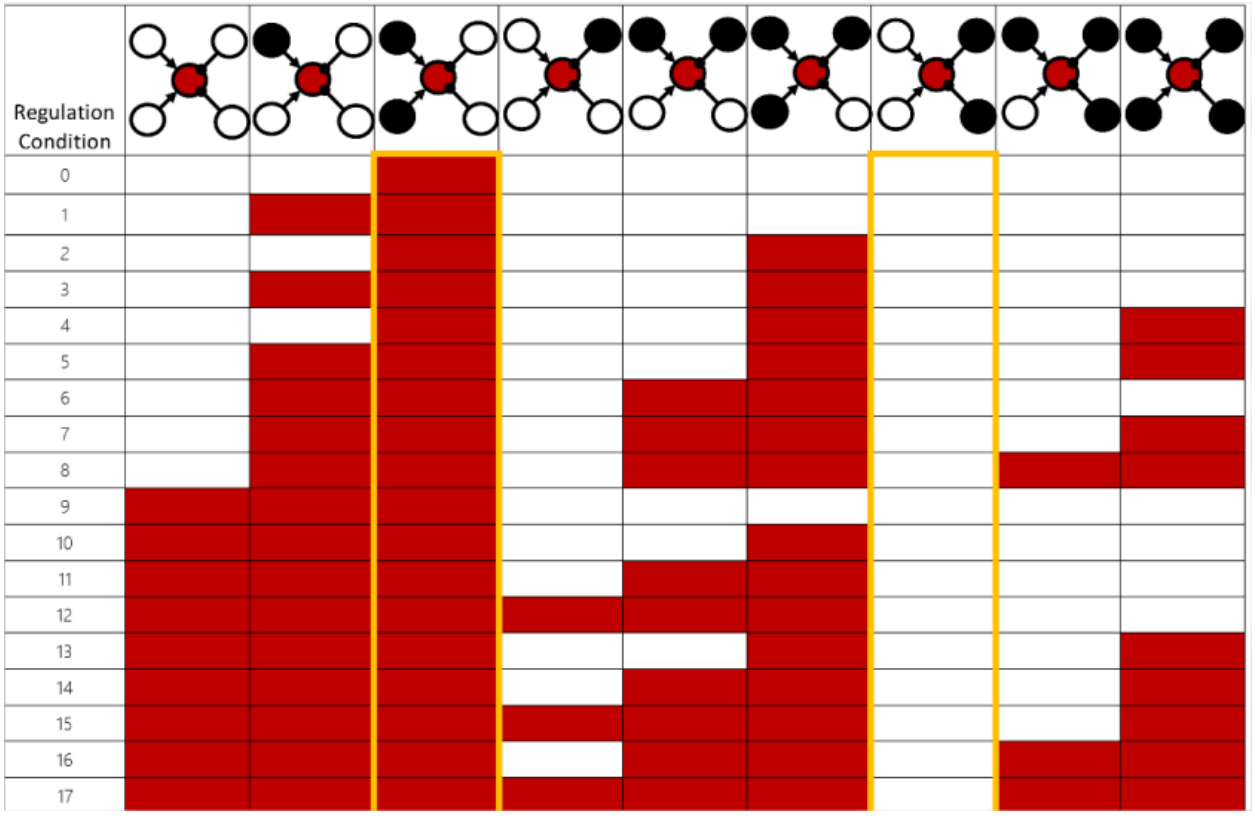
איליה ליפובן


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.

עבור המקרים הבאים:



נצפה לתוצאה הבאה:




נרצה לייצר את כל הפונקציות האפשריות תחילה
ולאחר מכן לבדוק אילו מהפונקציות מקיימות מונוטוניות.
נייצר 512 פונקציות שונות שמייצגות את כל הקומבינציות עם הפונקציות הבאות:

```python
# helper methods for getPossibleCombinations
def BinArrayToNum(arr):
    res = 0
    for idx, num in enumerate(arr[::-1]):
        res += num * math.pow(2, idx)
    return int(res)

def NumToBinArray(num, totallen):
    arr = []
    while num:
        res = num % 2
        arr.append(res)
        num = num // 2
    while totallen > len(arr):
        arr.append(0)
    return arr[::-1]
# returns all 2^9 possible functions
def getPossibleCombinations(length):
    res = []
    for i in range(2 ** length):
        res.append(NumToBinArray(i, length))
    return res
```

```python
# Filter functions that are monotonic
monoFunctions = []
for f in allFunctions:
    if checkMonotonicFunction(f):
        monoFunctions.append(f)
```

לאחר שייצרנו את כל הקומבינציות האפשריות של התוצאה, נרצה לבדוק עבור האפשרויות של הכניסהXמוצא אילו מונוטוניות.
הפונקציה הבאה מנסה לפסול את כל מה שלא מונוטוני ואם הכל תקין מחזירה TRUE.
מונוטוניות:
אם X<=Y->FX<=FY , כאשר X וY אלו האפשרויות השונות לINHIBITORS וACTIVATORS והFX וFY זה התוצאות הבוליאניות האפשריות.
נפסול את כל מה שלא מונוטוני על ידי התנאי : X<=Y and FX>FY
X<=Y כאשר מספר האקטיבטורים בY גדול מX או להפך עבור האינהיביטורים

נוסיף עוד 2 מקרי קצה לבדיקה: הפונקציה חייבת לפלוט 1 כאשר כל האקטיבטורים דולקים ו 0 במקרה שהאינהיביטורים דולקים.
סה"כ:

```python
def checkMonotonicFunction(function):
    for i in range(9):
        for j in range(9):
            a = allPossibilities[i]
            b = allPossibilities[j]
            #not(x>=y ->fx>=fy) = x>=y and fx<fy
            # same inhibitors but activators are greater or equal in number and the value
            c1_inh = a[2] + a[3]
            c2_inh = b[2] + b[3]
            c1_act = a[0] + a[1]
            c2_act = b[0] + b[1]
            if c1_inh == c2_inh and c1_act<c2_act and function[i] > function[j]:
                return False
            # same activators but inhibitors are greater or equal in number and the value
            if c1_act == c2_act and c1_inh>c2_inh and function[i] > function[j]:
                return False
    # if all inhibitors or activators are on:
    if function[2] != 1 or function[6] != 0:
        return False
    return True
```

אחרי הרצה נקבל:

| act:0,0 inh:0,0 | act:1,0 inh:0,0 | act:1,1 inh:0,0 | act:0,0 inh:1,0 | act:1,0 inh:1,0 | act:1,1 inh:1,0 | act:0,0 inh:1,1 | act:1,0 inh:1,1 | act:1,1 inh: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 |

כדרוש.

עבדנו עם קובץ utils.py ו findMonotonic.py הקובץ הראשון מכיל את הפונקציות להרצה והשני מריץ את החישוב ושומר את התוצאה כתמונה + PRINTOUT.

PRINTOUT:

```
[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]
```

תודה רבה!