FAF.FIA16.1 -- Artificial Intelligence Fundamentals

> **Lab 1:** Expert Systems
> **Performed by:** Furdui Alexandru, group FAF-192
> **Verified by:** Mihail Gavrilita, asist. univ.

Imports and Utils

```
from production import forward_chain
import rules
from production import OR, AND, match, populate, simplify
from production import IF, AND, THEN
import sys
from collections import UserDict
import regex as re
```

+ Cod + Text

Task 1 -- Define 5 types of tourists that visit Luna-City. Draw the Goal Tree representing these

types of tourists.

Task 3 -- If you are using the

```
tree = Tree('{0} has backpack and camera',
            Tree('{0} knows who his father is',
                Tree('{0} has bachelor')
                ),
            Tree('{0} doesnt know who his father is',
                Tree('{0} sells drugs',
                    Tree('{0} has dark past')
                    )
                ),
            Tree('{0} knows the difference between shaorma, doner and kebab'
                Tree('{0} wears fez',
                    Tree('{0} hates kurdish')
                    )
                ),
            Tree('{0} loves guacamole',
                Tree('{0} dances while making salsa',
                     Tree('{0} immigrated to the US')
                     )
                 ),
            Tree('{0} eats bats',
                Tree('{0} started covid',
                    Tree('{0} says nothing happened on 5th of June, 1989')
                    )
                )
            )
```

This is a simple Vector class that uses NumPy lib to implement vector operations

Task 2 -- Implement the rules from the defined tree

```
r
o
v
i
d
e
d
c
o
d
e
,
c
h
e
c
k
h
o
w
t
h
e
F
o
r
w
```

```
RULES = (
    IF(AND('(?y) has backpack and camera'),  # tourist
       THEN('(?y) is a tourist')),

    IF(AND('(?y) is a tourist',  # educated black tourist
           '(?y) knows who his father is',
           '(?y) has bachelor'),
       THEN('(?y) is an educated black tourist')),

    IF(AND('(?y) is a tourist',  # naughty black tourist
           '(?y) doesnt know who his father is',
           '(?y) sells drugs',
           '(?y) has dark past'),
       THEN('(?y) is a naughty black tourist')),

    IF(AND('(?y) is a tourist',  # chinesse tourist
           '(?y) eats bats',
           '(?y) started covid',
           '(?y) says nothing happened on 5th of June, 1989'),
       THEN('(?y) is a chinesse tourist')),

    IF(AND('(?y) is a tourist',  # turkish tourist
           '(?y) knows the difference between shaorma, doner and kebab',
           '(?y) wears fez',
           '(?y) hates kurdish'),
       THEN('(?y) is a turkish tourist')),

    IF(AND('(?y) is a tourist',  # mexican tourist
           '(?y) loves guacamole',
           '(?y) dances while making salsa',
           '(?y) immigrated to the US'),
       THEN('(?y) is a mexican tourist')),
)
```

Task 3 -- If you are using the provided code, check how the Forward Chaining algorithm works and illustrate an example

```
give name
Name: tim
Forward or Backward chain?(F/B):F
tim has backpack and camera? y/n y
tim knows who his father is? y/n y
tim has bachelor? y/n y
('tim has bachelor', 'tim has backpack and camera', 'tim is a tourist', 'tim
```

a r d C h a i n i n g a l g o r i t h n w o r k s a n

based on the input and the rules, we get all the type of tourist along with all the rules from where resulted the conclusion

Task 4 -- Implement the Backward Chaining algorithm for the Goal Tree.

```
def backchain_to_goal_tree(rules, hypothesis):
    length = len(rules)

    if length==0:
        return hypothesis
    tree = AND()

    for element in rules:
        con = element.consequent()
        mat = match(con[0], hypothesis)
        if mat is not None and len(mat)>=0:
            antec = element.antecedent()
            if isinstance(antec, list):
                sub = AND()
                if isinstance(antec, OR): sub = OR()
                for x in antec:
                    new_tree = backchain_to_goal_tree(rules, populate(x, mat
                    sub.append(new_tree)
                tree.append(sub)
            else:
                new_tree = backchain_to_goal_tree(rules, populate(antec, mat
                tree.append(AND(new_tree))
        else:
            tree.append(hypothesis)
    new = simplify(tree)
    return new
```

Task 5 -- Implement a system for generating questions from the Goal Tree

d i l l u s t r a t e a n e x a n p l e

```
def ask_questions(tree, name):
    if tree is None:
        return False

    answer = input(tree.cargo.format(name) + '? y/n ')
    while answer not in ['y', 'n']:
        answer = input("Please enter 'y' or 'n': ")

    subtree_explored = False
    if answer == 'y':
        for subtree in tree.subtrees:
            if ask_questions(subtree, name):
                answers.append(tree.cargo.format(name))
                return True
            if answers and answers[-1] == tree.cargo.format(name):
                subtree_explored = True
        if not subtree_explored and not tree.subtrees:
            answers.append(tree.cargo.format(name))
            return True
    elif tree.right:
        if ask_questions(tree.right, name):
            return True
    return False
```

Task 6 -- Wrap up everything in an interactive Expert System

```
Give name
Name: tim
Forward or Backward chain?(F/B):F
tim has backpack and camera? y/n y
tim knows who his father is? y/n n
tim doesnt know who his father is? y/n n
tim knows the difference between shaorma, doner and kebab? y/n n
tim loves guacamole? y/n n
tim eats bats? y/n y
tim started covid? y/n y
tim says nothing happened on 5th of June, 1989? y/n y
('tim eats bats', 'tim has backpack and camera', 'tim is a chinesse tourist'
tim is a chinesse tourist
tim is a tourist
It is not a tourist
Give name
Name: tim
Forward or Backward chain?(F/B):B
What type of tourist are you? chinese
AND('(?y) is a chinesse tourist', '(?y) has backpack and camera', '(?y) is a
```

Conclusions:

Concluding, after implementing this laboratory work, i undestood the basisc of the reasoning by implementing the bacward chain and understanding the forward chaining

Bibliography:

https://www.section.io/engineering-education/forward-and-backward-chaining-in-ai/ - Forward and Backward Chaining in Artificial Intelligence