Technical University of Moldova

Faculty of Computers, Informatics and Microelectronics

Department of Software Engineering and Automation

Fundamentals of Artificial Intelligence

Laboratory work #1

Expert Systems

Author:
Popa Eugeniu
std. gr. FAF-202

Supervisor:
Diana Marusic

Chișinău 2023

1 Task 1

Define 5 types of tourists. Draw the Goal Tree representing these types of tourists.

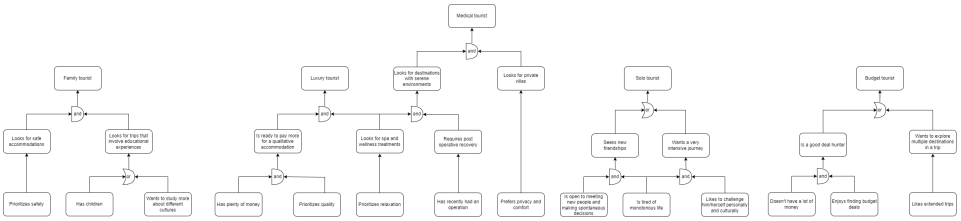

Figure 1: Goal Tree

2 Task 2

Implement the rules from the defined tree in task 1 in your code (use the IF, AND, OR, THEN etc rules which are already implemented in the code).

```
from production import IF, AND, THEN, OR


class FamilyRules:
    safety = "(?x) prioritizes safety"
    children = "(?x) has children"
    cultures = "(?x) wants to study more about different cultures"

    safe_accommodation = "(?x) looks for safe accommodations"
    education = "(?x) looks for trips that involve educational experiences"

    conclusion = "(?x) is a family tourist"

    def add(self):
        return [self.safety, self.children, self.cultures]


class CommonRules:
    relaxation = "(?x) prioritizes relaxation"

    treatments = "(?x) looks for spa and wellness treatments"

    def add(self):
        return [self.relaxation]


class LuxuryRules(CommonRules):
    plenty_of_money = "(?x) has plenty of money"
    quality = "(?x) prioritizes quality"

    qualitative_accommodation = "(?x) is ready to pay more for a qualitative
    accommodation"

    conclusion = "(?x) is a luxury tourist"

    def add(self):
        return [self.plenty_of_money, self.quality,
                self.relaxation]
```

```
38
39
40 class MedicalRules(CommonRules):
41     operation = "(?x) has recently had an operation"
42     privacy_and_comfort = "(?x) prefers privacy and comfort"
43
44     recovery = "(?x) requires post operative recovery"
45     serene_environments = "(?x) looks for destinations with serene environments"
46     private_villas = "(?x) looks for private villas"
47
48     conclusion = "(?x) is a medical tourist"
49
50     def add(self):
51         return [self.operation, self.privacy_and_comfort, self.relaxation]
52
53
54 class SoloRules:
55     monotony = "(?x) is tired of monotonous life"
56     socialization = "(?x) is open to meeting new people and making spontaneous
       decisions"
57     challenge = "(?x) likes to challenge him/herself personally and culturally"
58
59     friendships = "(?x) seeks new friendships"
60     intensity = "(?x) wants a very intensive journey"
61     conclusion = "(?x) is a solo tourist"
62
63     def add(self):
64         return [self.monotony, self.socialization, self.challenge]
65
66
67 class BudgetRules:
68     no_money = "(?x) doesn't have a lot of money"
69     budget_deals = "(?x) enjoys finding budget deals"
70     extended_trips = "(?x) likes extended trips"
71
72     deal_hunter = "(?x) is a good deal hunter"
73     multiple_destinations = "(?x) wants to explore multiple destinations in a trip"
74
75     conclusion = "(?x) is a budget tourist"
76
77     def add(self):
78         return [self.no_money, self.budget_deals, self.extended_trips]
79
80
81 TOURIST_RULES = (
82     # family tourist
83     IF(AND(FamilyRules.safety),
84        THEN(FamilyRules.safe_accommodation)),
85
86     IF(OR(FamilyRules.children, FamilyRules.cultures),
87        THEN(FamilyRules.education)),
88
89     IF(AND(FamilyRules.safe_accommodation, FamilyRules.education),
90        THEN(FamilyRules.conclusion)),
91
92     # common rule
93     IF(AND(CommonRules.relaxation),
94        THEN(LuxuryRules.treatments, MedicalRules.treatments)),
95
96     # luxury tourist
```

```
    IF(AND(LuxuryRules.plenty_of_money, LuxuryRules.quality),
       THEN(LuxuryRules.qualitative_accommodation)),

    IF(AND(LuxuryRules.relaxation),
       THEN(LuxuryRules.treatments)),

    IF(AND(LuxuryRules.qualitative_accommodation, LuxuryRules.treatments),
       THEN(LuxuryRules.conclusion)),

    # medical tourist
    IF(AND(MedicalRules.operation),
       THEN(MedicalRules.recovery)),

    IF(AND(MedicalRules.privacy_and_comfort),
       THEN(MedicalRules.private_villas)),

    IF(AND(MedicalRules.recovery, MedicalRules.treatments),
       THEN(MedicalRules.serene_environments)),

    IF(AND(MedicalRules.private_villas, MedicalRules.serene_environments),
       THEN(MedicalRules.conclusion)),

    # solo tourist
    IF(AND(SoloRules.monotony, SoloRules.socialization),
        THEN(SoloRules.friendships)),

    IF(AND(SoloRules.challenge, SoloRules.monotony),
       THEN(SoloRules.intensity)),

    IF(OR(SoloRules.friendships, SoloRules.intensity),
       THEN(SoloRules.conclusion)),

    # budget tourist
    IF(AND(BudgetRules.no_money, BudgetRules.budget_deals),
       THEN(BudgetRules.deal_hunter)),

    IF(AND(BudgetRules.budget_deals),
       THEN(BudgetRules.multiple_destinations)),

    IF(AND(BudgetRules.extended_trips),
       THEN(BudgetRules.multiple_destinations)),

    IF(OR(BudgetRules.deal_hunter, BudgetRules.multiple_destinations),
       THEN(BudgetRules.conclusion))
)
```

3 Task 3

If you are using the provided code, check how the Forward chaining algorithm works and show an example. If you are implementing your own code, implement the Forward chaining algorithm yourself.

```
TOURIST_DATA = (
    BudgetRules.budget_deals.replace("(?x)", "Eugeniu"),
)

print(forward_chain(TOURIST_RULES, TOURIST_DATA))
```

```
('Eugeniu enjoys finding budget deals', 'Eugeniu wants to explore multiple
    destinations in a trip')
```

In the provided code, there was an error stemming from the AIStringToRegex function in the utils.py file. This error message, specifically regex._regex_core.error: bad escape §at position 9, signaled a problem with an escape sequence in a regular expression pattern.

The issue arose because §was being interpreted as a special regular expression pattern, designed to match any non-whitespace character. However, in this context, §was meant to be taken as a literal string within the regular expression template, not as a pattern itself. To resolve this problem and treat §as a literal string, it was necessary to escape the backslash character because backslashes have special meanings in regular expressions.

The solution involved modifying the regular expression template by adding an extra backslash before §, like so:

```
def AIStringToRegex(AIStr):
    res = AIRegex.sub(r'(?P<\1>\\S+)', AIStr) + '$'
    return res
```

4 Task 4

Implement the Backward chaining algorithm for the Goal tree.

```
def backward_chain(rules, hypothesis, list_rules=None, verbose=False):
    """
    Outputs the goal tree from having rules and hypothesis,
    works like an "encyclopedia"
    """

    if list_rules is None:
        list_rules = []

    for rule in rules:
        goal = match(rule.consequent()[0], hypothesis)
        if goal:
            list_rules.append(rule.antecedent())
            for antecedent in rule.antecedent():
                hypothesis = populate(antecedent, goal)
                backward_chain(rules, hypothesis, list_rules, verbose)

    return list_rules
```

```
AND('(?x) is ready to pay more for a qualitative accommodation', '(?x) looks for spa
    and wellness treatments'), AND('(?x) has plenty of money', '(?x) prioritizes
    quality'), AND('(?x) prioritizes relaxation')
```

5 Task 5

Implement a system for generating questions from the goal tree. Have at least 2-3 types of questions (e.g yes/no, multiple choice, etc).

```
class Choices:
    def __init__(self):
        self.solo_tourist_questions = SoloRules().add()
```

```
        self.family_tourist_questions = FamilyRules().add()
        self.budget_tourist_questions = BudgetRules().add()
        self.luxury_tourist_questions = LuxuryRules().add()
        self.medical_tourist_questions = MedicalRules().add()
        self.conclusions = [SoloRules.conclusion,
                            FamilyRules.conclusion,
                            BudgetRules.conclusion,
                            LuxuryRules.conclusion,
                            MedicalRules.conclusion
                            ]

    def combine_questions(self):
        return list(
            set(self.solo_tourist_questions +
                self.family_tourist_questions +
                self.budget_tourist_questions +
                self.luxury_tourist_questions +
                self.medical_tourist_questions
                )
        )

    def forward(self, name):
        print(f"Choose the facts about you:")
        facts = ""
        for index, fact in enumerate(self.combine_questions()):
            fact = fact.replace("(?x)", "")
            answer = input(f"{index + 1} {fact} (yes/any key to continue)\n")
            if answer == "yes":
                facts = facts + " " + str(index)
                print(facts)
        print("-------------------------------")
        chain = []
        for index in facts.split(" "):
            try:
                fact_index = int(index) - 1
                fact = self.combine_questions()[fact_index]
                chain.append(fact.replace("(?x)", name))
            except (ValueError, IndexError):
                continue
        chained_data = forward_chain(TOURIST_RULES, chain)

        hints = []
        for conclusion in self.conclusions:
            dec_conclusion = conclusion.replace("(?x)", name)
            if dec_conclusion in chained_data:
                hints.append(dec_conclusion)
        if not hints:
            return f"{name} might be a Loonie."
        hints_str = ', '.join(map(str, hints))
        return hints_str

    def backward(self, name):
        print(f"Choose the tourist type from the list to execute backward chaining: "
    )
        for index, conclusion in enumerate(self.conclusions):
            conclusion = conclusion.replace("(?x)", name)
            print(index + 1, conclusion)
        selected = int(input("Choose: \n"))
        print("-------------------------------")
        if 0 <= selected < 5:
```

```
            goal = (self.conclusions[int(selected) - 1]).replace("(?x)", name)
            backward_chain_result = backward_chain(TOURIST_RULES, goal)
            backward_chain_result_str = ', '.join(map(str, backward_chain_result))
            return backward_chain_result_str
        else:
            return f"There is no such type of tourist."
```

6 Task 6

Wrap up everything in an Interactive Expert System that will dynamically ask questions based on the input from the user. Both forward chaining and backward chaining should be working.

```
if __name__ == '__main__':
    print("Welcome to Expert System!")
    print("————————————————————————————")
    print("Let's find your tourist type.")
    print("————————————————————————————")

    choices = Choices()

    user_name = input("Please, write your name: \n")
    print("————————————————————————————")
    print("Hello, " + user_name + "!")
    print("————————————————————————————")

    while True:
        print("Choose the algorithm you want to use: ")
        algorithm = input("1 forward chaining\n2 backward chaining\n")
        print("————————————————————————————")
        if algorithm == "1":
            print(choices.forward(user_name))
        elif algorithm == "2":
            print(choices.backward(user_name))
        else:
            print("Please, write a valid number")
        print("————————————————————————————")
        exit_command = input("Do you want to continue? (write yes/any key to exit the
    program):\n")
        print("————————————————————————————")
        if exit_command != "yes":
            print("Exiting...")
            break
```

7 Task 7

Format the output and questions to human readable format.

```
Welcome to Expert System!
————————————————————————————
Let's find your tourist type.
————————————————————————————
Please, write your name:
Eugen
————————————————————————————
Hello, Eugen!
————————————————————————————
```

```
Choose the algorithm you want to use:
1 forward chaining
2 backward chaining
1
-----------------------------------
Choose the facts about you:
1  prioritizes quality (yes/any key to continue)
yes
2  has recently had an operation (yes/any key to continue)
no
3  is tired of monotonous life (yes/any key to continue)
no
4  prioritizes safety (yes/any key to continue)
yes
5  has children (yes/any key to continue)
yes
6  doesn't have a lot of money (yes/any key to continue)
no
7  is open to meeting new people and making spontaneous decisions (yes/any key to
   continue)
yes
8  enjoys finding budget deals (yes/any key to continue)
no
9  has plenty of money (yes/any key to continue)
yes
10  wants to study more about different cultures (yes/any key to continue)
yes
11  likes extended trips (yes/any key to continue)
no
12  prioritizes relaxation (yes/any key to continue)
yes
13  likes to challenge him/herself personally and culturally (yes/any key to continue
   )
no
14  prefers privacy and comfort (yes/any key to continue)
yes
-----------------------------------
Eugen is a solo tourist, Eugen is a budget tourist
-----------------------------------
Do you want to continue? (write yes/any key to exit the program):
yes
-----------------------------------
Choose the algorithm you want to use:
1 forward chaining
2 backward chaining
1
-----------------------------------
Choose the facts about you:
1  prioritizes quality (yes/any key to continue)
yes
2  has recently had an operation (yes/any key to continue)
no
3  is tired of monotonous life (yes/any key to continue)
no
4  prioritizes safety (yes/any key to continue)
no
5  has children (yes/any key to continue)
no
6  doesn't have a lot of money (yes/any key to continue)
no
```

```
7  is open to meeting new people and making spontaneous decisions (yes/any key to
    continue)
yes
8  enjoys finding budget deals (yes/any key to continue)
no
9  has plenty of money (yes/any key to continue)
no
10  wants to study more about different cultures (yes/any key to continue)
no
11  likes extended trips (yes/any key to continue)
no
12  prioritizes relaxation (yes/any key to continue)
no
13  likes to challenge him/herself personally and culturally (yes/any key to continue
    )
no
14  prefers privacy and comfort (yes/any key to continue)
no
----------------------------------
Eugen might be a Loonie.
----------------------------------
Do you want to continue? (write yes/any key to exit the program):
yes
----------------------------------
Choose the algorithm you want to use:
1 forward chaining
2 backward chaining
2
----------------------------------
Choose the tourist type from the list to execute backward chaining:
1 Eugen is a solo tourist
2 Eugen is a family tourist
3 Eugen is a budget tourist
4 Eugen is a luxury tourist
5 Eugen is a medical tourist
Choose:
2
----------------------------------
AND('(?x) looks for safe accommodations', '(?x) looks for trips that involve
    educational experiences'), AND('(?x) prioritizes safety'), OR('(?x) has children',
     '(?x) wants to study more about different cultures')
----------------------------------
Do you want to continue? (write yes/any key to exit the program):
yes
----------------------------------
Choose the algorithm you want to use:
1 forward chaining
2 backward chaining
2
----------------------------------
Choose the tourist type from the list to execute backward chaining:
1 Eugen is a solo tourist
2 Eugen is a family tourist
3 Eugen is a budget tourist
4 Eugen is a luxury tourist
5 Eugen is a medical tourist
Choose:
6
----------------------------------
There is no such type of tourist.
```

```
124 ----------------------------------
125 Do you want to continue? (write yes/any key to exit the program):
126 e
127 ----------------------------------
128 Exiting...
129
130 Process finished with exit code 0
```

8 Conclusion

In the course of conducting this laboratory work, I have acquired valuable insights into the realm of AND/OR-trees, forward and backward chaining techniques, and the fundamental principles governing Rule-Based Expert Systems. In summary, although my system may be less intricate than a full-fledged Expert system, it fundamentally operates on rule-based logic and forward chaining principles. This experience has proven to be an excellent foundation for further, more substantial projects in the future, rendering it a noteworthy and instructive learning journey with the potential to serve as a springboard for more advanced endeavors.