МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РФ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ «ЛИПЕЦКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

Институт компьютерных наук
Кафедра автоматизированных систем управления

ЛАБОРАТОРНАЯ РАБОТА №2

По предмету: “СИСТЕМЫ ИСКУССТВЕННОГО ИНТЕЛЛЕКТА”

Разработка экспертной системы с использованием обратной цепочки рассуждений

Студент АС-21-1 _____ Станиславчук С.М.

(подпись, дата)

Руководитель

Профессор _____ Сараев П.В.

(подпись, дата)

Липецк, 2024 г.

Цель работы

Получение навыков проектирования и разработки экспертной системы на всех этапах ее создания.

Задача: разработать экспертную систему, которая поможет начинающему разработчику выбрать подходящий язык программирования для его проекта. Система должна учитывать различные факторы, такие как тип проекта, требования к производительности, простота использования, поддержка сообщества и другие критерии.

Задание кафедры

1. Разработайте оболочку экспертной системы, реализующую обратную цепочку рассуждений, на любом языке программирования.
2. Требования к базе знаний аналогичны требованиям в лабораторной работе № 1.
3. Задайте начальное состояние рабочей базы данных.
4. Задайте целевую ситуацию.
5. С помощью обратной цепочки рассуждений докажите или опровергните возможность достижения заданной целевой ситуации.
6. Выберите предметную область из лабораторной работы № 1 или предложите другую, для которой составьте базу правил.
7. Протестируйте работу вашей программы на этой базе правил.

Допускается использование пакетов языка Python для составления базы правил и проведения логического выводов.

База правил:

ЕСЛИ область_разработки=веб-разработка И совместимость=низкая_важность ТО приоритет=JavaScript

ЕСЛИ скорость=важна И время_разработки=не_важно ТО приоритет=C++

ЕСЛИ скорость=важна И область_разработки=драйверы ТО приоритет=C

ЕСЛИ скорость=не_важна И область_разработки=математика ТО приоритет=Python

ЕСЛИ область_разработки=мобильные_приложения И совместимость=высокая_важность ТО приоритет=Kotlin

ЕСЛИ скорость=важна И область_разработки=игры ТО приоритет=C#

ЕСЛИ время_разработки=важно И область_разработки=научные_исследования ТО приоритет=R

ЕСЛИ совместимость=низкая_важность И область_разработки=встраиваемые_системы ТО приоритет=Rust

ЕСЛИ область_разработки=десктопные_приложения И совместимость=высокая_важность ТО приоритет=Java

ЕСЛИ скорость=важна И область_разработки=обработка_данных ТО приоритет=Go

ЕСЛИ время_разработки=не_важно И область_разработки=автоматизация ТО приоритет=Shell

ЕСЛИ совместимость=низкая_важность И область_разработки=искусственный_интеллект ТО приоритет=Julia

ЕСЛИ скорость=важна И область_разработки=веб-разработка ТО приоритет=TypeScript

ЕСЛИ скорость=не_важна И область_разработки=разработка_программного_обеспечения ТО приоритет=PHP

ЕСЛИ время_разработки=важно И область_разработки=облачные_технологии ТО приоритет=Scala

Программная реализация

```
import subprocess
from rich.columns import Columns
from rich.console import Console
from rich.panel import Panel
from rich.prompt import Prompt
from rich.table import Table


def load_rules(_file_path):
    rules = []
    with open(_file_path, "r", encoding="utf-8") as file:
        for line in file:
            if line.strip():
                condition, result = line.strip().split(" ТО ")
                conditions = condition.replace("ЕСЛИ ", "").split(" И ")
                result_obj, result_val = result.split("=")
                rules.append((conditions, (result_obj.strip(),
result_val.strip())))
    return rules


def load_start_values(_file_path):
    facts = {}
    with open(_file_path, "r", encoding="utf-8") as file:
        for line in file:
            key, value = line.strip().split("=")
            facts[key] = value
    return facts


def reverse_inference(_rules, _trg, _facts):
    checked_goals = set()
    applied_rules = []

    def prove(goal_obj, trg_val):
        if (goal_obj, trg_val) in checked_goals:
            return False

        if _facts.get(goal_obj, "").lower() == trg_val.lower():
            return True

        for ifs, (res_obj, res_val) in _rules:
            if res_obj == goal_obj and res_val.lower() == trg_val.lower():
                all_conditions_met = True
                for condition in ifs:
                    cond_obj, cond_val = condition.split("=")
                    if not prove(cond_obj.strip(), cond_val.strip()):
                        all_conditions_met = False
                        break
                if all_conditions_met:
                    _facts[goal_obj] = trg_val
                    applied_rules.append(
                        f"ЕСЛИ {' И '.join(ifs)} ТО {res_obj} = {res_val}"
                    )
                    return True

        checked_goals.add((goal_obj, trg_val))
        return False
```

```
    success = prove(_trg[0], _trg[1])
    return success, applied_rules


def display_facts(_facts):
    table = Table(title="Facts")
    table.add_column("Object", justify="left")
    table.add_column("Value", justify="left")
    for obj, val in _facts.items():
        table.add_row(obj, val)
    return table


def display_target(_target):
    table = Table(title="Target")
    table.add_column("Object", justify="left")
    table.add_column("Value", justify="left")
    table.add_row(_target[0], _target[1])
    return table


def display_applied_rules(_applied_rules):
    table = Table(title="Applied Rules")
    table.add_column("Rule", justify="center")
    for rule in _applied_rules:
        table.add_row(rule)
    return table


def display_iteration_logs(_iteration_logs):
    table = Table(title="Iteration Logs")
    table.add_column("Log Entry", justify="left")
    for log in _iteration_logs:
        table.add_row(log)
    return table


def display_help():
    help_table = Table(title="Help Instructions")
    help_table.add_column("Action", justify="left")
    help_table.add_column("Description", justify="left")
    help_table.add_row("1", "Run Inference")
    help_table.add_row("2", "Add Fact")
    help_table.add_row("3", "Edit Fact")
    help_table.add_row("4", "Remove Fact")
    help_table.add_row("5", "Clear Facts")
    help_table.add_row("6", "Open Rules Editor")
    help_table.add_row("7", "Edit Target")
    help_table.add_row("8", "Exit")
    return help_table


def run():
    console = Console()
    rules_file = "bz.txt"
    target = ("бумага", "текст")

    rules = load_rules(rules_file)
    facts = load_start_values("start_val.txt")

    while True:
```

```
        console.clear()
        console.print(Panel("Expert System TUI"), justify="center")

        facts_table = display_facts(facts)
        target_table = display_target(target)
        help_table = display_help()

        columns = Columns([facts_table, target_table, help_table])
        console.print(columns)

        action = Prompt.ask(
            "Choose an action ",
            choices=["1", "2", "3", "4", "5", "6", "7", "8"],
        )

        if action == "1":
            applicable, applied_rules = reverse_inference(rules, target,
facts)
            if not applicable:
                console.print(
                    "No applicable rules. Please add new facts.",
style="bold red"
                )
            else:
                console.print(
                    f"'{target[0]} = {target[1]}' achieved!", style="bold
green"
                )
                print("\nApplied facts:")
                console.print(display_facts(facts))

                print("\nWe pass the following rules:")
                rule_count = 0
                for rule in applied_rules:
                    rule_count += 1
                    print(f"{rule_count}. {rule}")
                    input("Press Enter to continue...")

        elif action == "2":
            obj = Prompt.ask("Enter fact name")
            val = Prompt.ask("Enter fact value")
            if obj and val:
                facts[obj] = val
            else:
                console.print(
                    "Error: Both object and value must be provided.",
style="bold red"
                )

        elif action == "3":
            obj = Prompt.ask("Enter fact name to edit")
            if obj in facts:
                val = Prompt.ask("Enter new fact value")
                facts[obj] = val
            else:
                console.print("Error: Fact not found.", style="bold red")

        elif action == "4":
            obj = Prompt.ask("Enter fact name to remove")
            if obj in facts:
                del facts[obj]
            else:
```

```
                console.print("Error: Fact not found.", style="bold red")

        elif action == "5":
            facts.clear()

        elif action == "6":
            try:
                subprocess.call(["kitty -e nvim bz.txt"], shell=True)
            except FileNotFoundError:
                console.print("Error: rules.py file not found.", style="bold
red")

        elif action == "7":
            obj = Prompt.ask("Enter target object", default=target[0])
            val = Prompt.ask("Enter new target value", default=target[1])
            target = (obj, val)

        elif action == "8":
            break


if __name__ == "__main__":
    run()
```

Рисунок 1 — Пример выполнения программы

```
1   ЕСЛИ область_разработки=веб-разработка И совместимость=низкая_важность ТО приоритет=JavaS
    cript
  1 ЕСЛИ скорость=важна И время_разработки=не_важно ТО приоритет=C++
  2 ЕСЛИ скорость=важна И область_разработки=встраиваемые_системы ТО приоритет=C
  3 ЕСЛИ скорость=не_важна И область_разработки=математика ТО приоритет=Python
  4 ЕСЛИ область_разработки=мобильные_приложения И совместимость=высокая_важность ТО приорите
    т=Kotlin
  5 ЕСЛИ скорость=важна И область_разработки=игры ТО приоритет=C#
  6 ЕСЛИ время_разработки=важно И область_разработки=научные_исследования ТО приоритет=R
  7 ЕСЛИ совместимость=низкая_важность И область_разработки=встраиваемые_системы ТО приоритет
    =Rust
  8 ЕСЛИ скорость=не_важна И область_разработки=базы_данных ТО приоритет=SQL И удалить=совмес
    тимость
  9 ЕСЛИ область_разработки=десктопные_приложения И совместимость=высокая_важность ТО приорит
    ет=Java
 10 ЕСЛИ скорость=важна И область_разработки=обработка_данных ТО приоритет=Go
 11 ЕСЛИ время_разработки=не_важно И область_разработки=автоматизация ТО приоритет=Shell
 12 ЕСЛИ совместимость=низкая_важность И область_разработки=искусственный_интеллект ТО приори
    тет=Julia
 13 ЕСЛИ скорость=важна И область_разработки=веб-разработка ТО приоритет=TypeScript
 14 ЕСЛИ скорость=не_важна И область_разработки=разработка_программного_обеспечения ТО приори
    тет=PHP
 15 ЕСЛИ время_разработки=важно И область_разработки=облачные_технологии ТО приоритет=Scala
~
~
```

Рисунок 2 — База правил в отдельном текстовом файле .txt с приоритетами

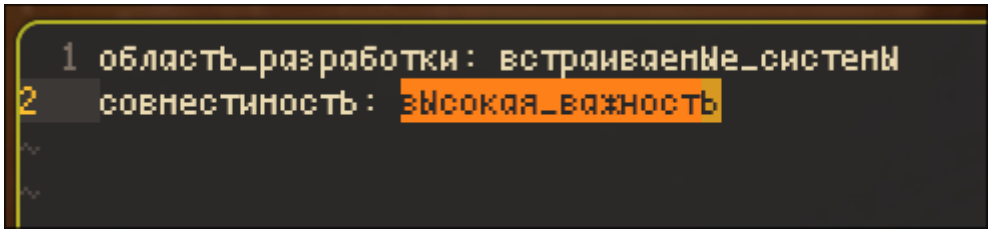

Рисунок 3 — Начальное состояние фактов

Вывод:

В ходе выполнения лабораторной работы получил навыки проектирования и разработки экспертной системы на всех этапах ее создания.