Федеральное агентство по образованию

Пермский национальный исследовательский политехнический университет

Кафедра вычислительной математики и механики

ЛАБОРАТОРНАЯ РАБОТА № 2

Проектирование и реализация оболочки экспертной

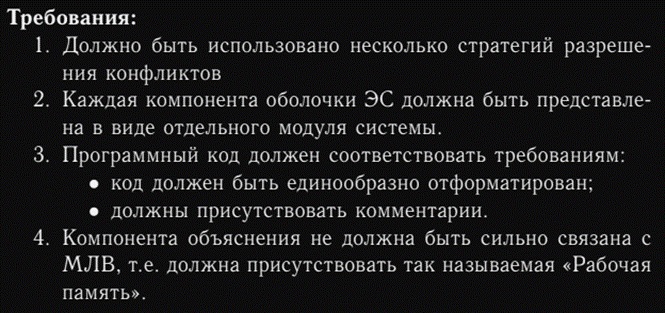
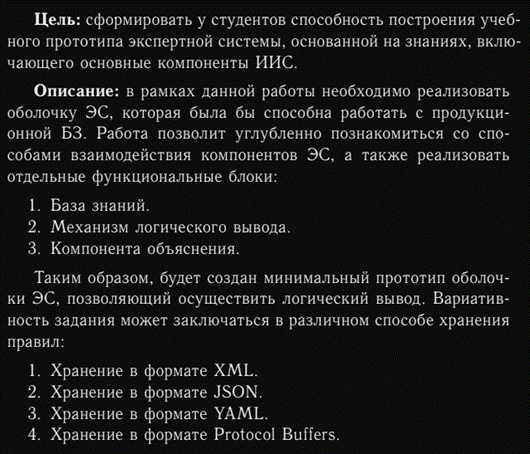
системы, использующей продукционный способ

представления знаний

Выполнил:

студент гр. ИСТ-19-2б А.А. Михайлишин

Пермь 2022



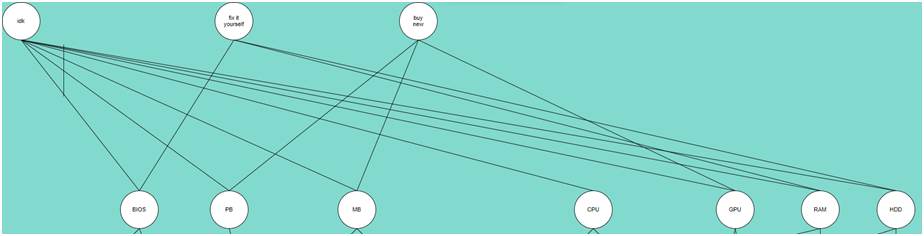
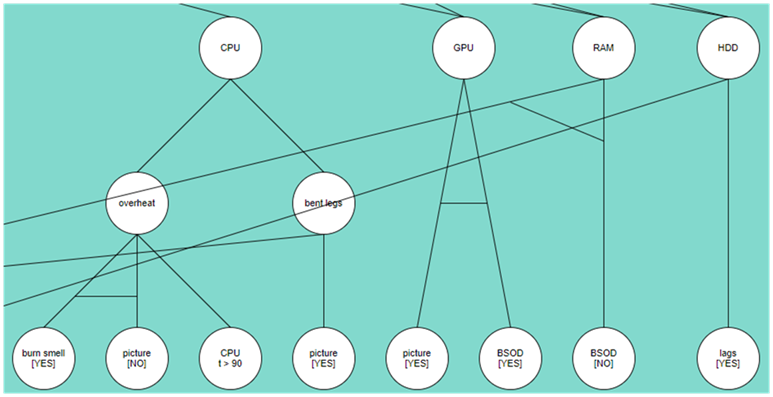
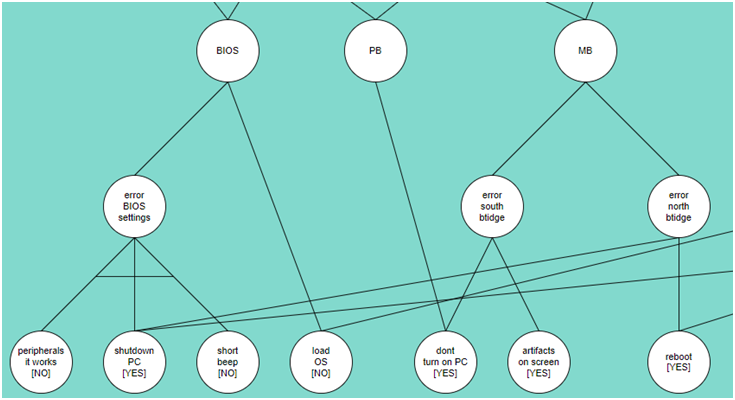
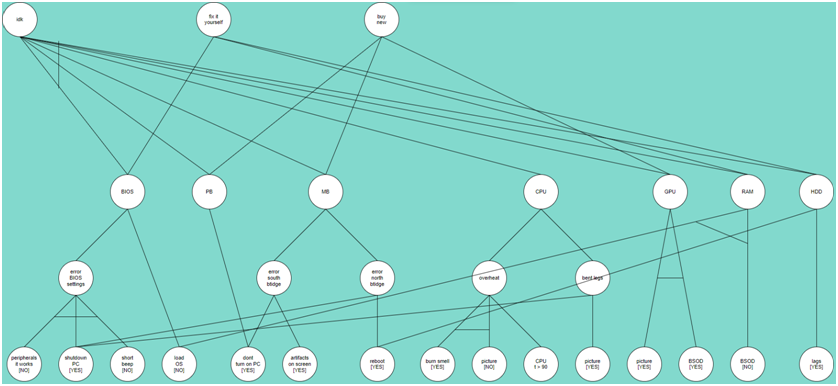
1. **Устройство БЗ в JSON**

В JSON хранится список ([…]) правил ({…}). Рассмотрим одно из правил:



* Ключ “name”, соответствует названию правила.
* Ключ “condition”, объявляет блок с условиями (в том числе вложенные условия).
* Ключ “result”, объявляет блок действий при выполнение условия из “condition”.
* Ключ “method”, соответствует названию метода, который нужно использовать.
* Ключ “arguments”, соответствует параметрам, необходимым для выполнения “method.”

**2. Визуально оформленный граф**



**3. Код оболочки на языке Python**

import json  
  
  
def And(\*params):  
 return all(params)  
  
  
def AskValue(question):  
 while True:  
 answer = input(question)  
 if answer.isdigit():  
 break  
 return answer  
  
  
def Biggest(param, value):  
 if not (param in variables):  
 variables[param] = 0  
 return float(variables[param]) > float(value)  
  
  
def Compare(param, value):  
 if not (param in variables):  
 variables[param] = 0  
 return variables[param] == value  
  
  
def Or(\*params):  
 return any(params)  
  
  
def Print(text):  
 print(text)  
  
  
def SetValue(param, value):  
 variables[param] = value  
 return variables[param]  
  
  
def YesOrNo(question):  
 answer = ""  
  
 while answer != "yes" and answer != "no":  
 answer = input(question)  
  
 if answer == "y":  
 answer = "yes"  
 if answer == "n":  
 answer = "no"  
 return answer  
  
  
methods = {"and": And, "askValue": AskValue, "biggest": Biggest, "compare": Compare, "or": Or, "print": Print, "setValue": SetValue, "yesOrNo": YesOrNo}  
variables = {"solution": 0}  
  
  
with open("rules.json", "r", encoding="utf-8") as rules:  
 data = json.load(rules)  
  
  
def ruleManager(part):  
 total\_arguments = []  
 if isinstance(part["arguments"], str):  
 total\_arguments.append(part["arguments"])  
 else:  
 for arg in part["arguments"]:  
 if not (isinstance(arg, dict)):  
 total\_arguments.append(arg)  
 else:  
 total\_arguments.append(ruleManager(arg))  
 return methods.get(part["method"])(\*total\_arguments)  
  
  
def ruleHandler():  
 i = len(data) - 1  
 skip = []  
 while variables["solution"] == 0 and i > 0:  
 if i not in skip:  
 if ruleManager(data[i]["condition"]):  
 ruleManager(data[i]["result"])  
 skip.append(i)  
 i = len(data)  
 i -= 1  
  
  
ruleHandler()