МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«Санкт-Петербургский национальный исследовательский университет информационных технологий, механики и оптики»

ФАКУЛЬТЕТ ПРОГРАММНОЙ ИНЖЕНЕРИИ И КОМПЬЮТЕРНОЙ ТЕХНИКИ

**Отчёт по модулю №1**

по дисциплине

«Системы искусственного интеллекта»

***Выполнил:*** Студент группы P3334 Баянов Равиль Динарович

***Преподаватель:***

Авдюшина А. Е.

Санкт-Петербург, 2024

**Оглавление**

[**Введение** 3](#_Toc178113988)

[**Анализ требований** 4](#_Toc178113989)

[**Изучение основных концепций и инструментов** 5](#_Toc178113990)

[**Реализация системы искусственного интеллекта (системы поддержки принятия решения)** 6](#_Toc178113991)

[**Оценка и интерпретация результатов** 11](#_Toc178113992)

[**Вывод** 14](#_Toc178113993)

**Введение**

В первых двух лабораторных работах мы столкнулись с изучением истоков систем искусственного интеллекта. Начало ИИ строится на основе логических рассуждений. И первая практическая идея построения интеллектуальных систем была реализована на декларативном языке программирования Prolog, построенном на основе логики, фактов и правил. Первая часть первой лабораторной работы продвигает цель научить нас применять идеи последовательных логических действий на языке Prolog. Познакомившись с предикатами и синтаксисом Prolog, мы хотим научиться определить для себя границы применимости систем искусственного интеллекта и научиться строить простейшие базы знаний. Вторая же часть знакомит нас с понятием онтологий и их значимостью в сфере ИИ. Благодаря программе Protégé, мы научимся создавать примитивные онтологии, на основе которых в будущем могут строиться системы искусственного интеллекта. Ну и визуализировать полученные знания нам поможет вторая лабораторная работа, которая в свою очередь предлагает нам уже построенные нами базу знаний и онтологию реализовать на любом языке программирования в виде рекомендательной системы. Первый модуль плавно знакомит нас с элементами систем ИИ, которые строятся на основе логики и на основе данных.

**Анализ требований**

Основные требования к системе поддержки принятия решений это:

* Точность и полнота
* Гибкость
* Ясность и удобство взаимодействия с системой

Рекомендательная система обязана быть быстрой и полной. Пользователь не должен получать ответы на свои запросы, которые будут не точными или лживыми. Именно поэтому такую систему нужно постоянно поддерживать и обновлять, чтобы такая система всегда выдавала актуальный ответ.

Основные требования к БЗ и онтологии:

* Плотная связь онтологии и БЗ
* Структурированность и логичность
* Чёткость и ясность субъектов, объектов и отношений (предикатов)

Базы знаний и онтологии — это неразрывные вещи. Абсолютно все онтологии (семантические сети и паутины) строятся на основе какой-то определённой БЗ. Именно поэтому не должно быть никаких расхождений. Также онтология и БЗ должны быть чётко структурированы и удобочитаемы. Иначе люди и другие системы, пользующиеся этими онтологиями, могут не так интерпретировать данные. Онтологии для этого и созданы, чтобы укомплектовать какую-то предметную область в приятную глазу концептуальную схему.

**Изучение основных концепций и инструментов**

Базы знаний и онтологии — это два важных подхода к организации знаний в системах ИИ. Базы знаний базируются на основе хранения данных и связей между этими данными на основе предикатов, правил и фактов. Фактами являются элементарными единицами в создании баз знаний. Правила же собираются из фактов и создают более сложные знания в базе. Всё это собирается в логические выводы о предметной области. И это помогает нам в принятии решений в определённых ситуациях.

Онтологии помогают нам формализовать данные, хранящиеся в базе знаний с помощью схем, семантических сетей и графов. Онтологии базируются на классах, свойствах, отношениях и ограничениях (подобно объектно-ориентированному программированию). Чем сложнее БЗ, тем сильнее она зависима от её формализации. Но при увеличении базы знаний онтология может перестать быть читаемой.

Prolog – очень мощный язык декларативной парадигмы. Он оперирует предикатами, фактами и правилами. Он позволяет находить всевозможные решения логических задач, по-своему перебирая факты. Он выводит логические выводы по запросу из базы знаний. Prolog обладает рекурсией и массивами, что позволяет строить сложные правила, на основе которых можно искать интересные логические пути и применять их в создании систем ИИ. Prolog умеет сопоставлять факты между собой, сравнивать их и с помощью заглушек регулирует объём поиска нужных данных.

Для начала изучения Prolog поможет простой и понятный интерпретатор SWI-Prolog. Этот инструмент обладает OWL и RDF форматами, которые позволяют сохранять базы знаний в текстовом формате. С помощью RDF (расширения для XML) можно создать на основе данной БЗ онтологию. А с помощью SPARQL запросов можно обращаться в базе знаний, написанной на Prolog.

**Реализация системы искусственного интеллекта (системы поддержки принятия решения)**

Я реализовал СППР на языке программирования Java. Вся логика хранится в классе Data, которые является как бы хранилищем для фактов и правил. И приложение обращаясь к этой базе выдаёт пользователю ответ.

Код:

|  |
| --- |
| 1. **public** **class** Main { 2. **public** **static** **void** main([String](http://www.google.com/search?hl=en&q=allinurl%3Adocs.oracle.com+javase+docs+api+string)[] args) { 3. Supervisor supervisor = **new** Supervisor(); 4. supervisor.run(); 5. } 6. } 7. **public** **class** Main { 8. **public** **static** **void** main([String](http://www.google.com/search?hl=en&q=allinurl%3Adocs.oracle.com+javase+docs+api+string)[] args) { 9. Supervisor supervisor = **new** Supervisor(); 10. supervisor.run(); 11. } 12. }**import** model.Brawler; 13. **import** model.Data; 15. **import** java.util.Locale; 16. **import** java.util.Objects; 17. **import** java.util.Scanner; 19. **public** **class** Supervisor { 20. Scanner console = **new** Scanner([System](http://www.google.com/search?hl=en&q=allinurl%3Adocs.oracle.com+javase+docs+api+system).in); 21. Data data; 23. Supervisor() { 24. **this**.data = **new** Data(); 25. data.getArray().add(**new** Brawler("shelly", "common", 3000, 7400)); 26. data.getArray().add(**new** Brawler("nita", "rare", 1920, 8000)); 27. data.getArray().add(**new** Brawler("bull", "rare", 880, 10000)); 28. data.getArray().add(**new** Brawler("el\_primo", "rare", 760, 12000)); 29. data.getArray().add(**new** Brawler("colt", "rare", 4320, 5600)); 30. data.getArray().add(**new** Brawler("tick", "super\_rare", 1280, 4400)); 31. data.getArray().add(**new** Brawler("penny", "super\_rare", 1960, 6400)); 32. data.getArray().add(**new** Brawler("carl", "super\_rare", 1480, 8000)); 33. data.getArray().add(**new** Brawler("rico", "super\_rare", 640, 5600)); 34. data.getArray().add(**new** Brawler("bo", "epic", 1280, 7200)); 35. data.getArray().add(**new** Brawler("piper", "epic", 3400, 4600)); 36. data.getArray().add(**new** Brawler("bea", "epic", 1600, 5000)); 37. data.getArray().add(**new** Brawler("emz", "epic", 1040, 7200)); 38. data.getArray().add(**new** Brawler("tara", "mythic", 960, 6200)); 39. data.getArray().add(**new** Brawler("mortis", "mythic", 2000, 7600)); 40. data.getArray().add(**new** Brawler("buzz", "mythic", 840, 9600)); 41. data.getArray().add(**new** Brawler("max", "mythic", 640, 6600)); 42. data.getArray().add(**new** Brawler("chuck", "mythic", 1080, 9000)); 43. data.getArray().add(**new** Brawler("sandy", "legendary", 1800, 7600)); 44. data.getArray().add(**new** Brawler("leon", "legendary", 960, 6800)); 45. data.getArray().add(**new** Brawler("crow", "legendary", 640, 4800)); 46. data.getArray().add(**new** Brawler("surge", "legendary", 2360, 6800)); 47. data.getArray().add(**new** Brawler("kordelius", "legendary", 1400, 6400)); 48. data.getArray().add(**new** Brawler("amber", "legendary", 4200, 6400)); 49. data.getArray().add(**new** Brawler("spike", "legendary", 1080, 5200)); 50. data.getMap\_good\_team().put("max", "surge"); 51. data.getMap\_good\_team().put("tara", "sandy"); 52. data.getMap\_good\_team().put("sandy", "tara"); 53. data.getMap\_good\_team().put("surge", "max"); 54. } 56. *// Запуск приложения* 57. **void** run() { 58. [System](http://www.google.com/search?hl=en&q=allinurl%3Adocs.oracle.com+javase+docs+api+system).out.println("Добро пожаловать в базу знаний по мобильной игре Brawl Stars"); 59. **while** (**true**) { 60. [System](http://www.google.com/search?hl=en&q=allinurl%3Adocs.oracle.com+javase+docs+api+system).out.print(">>>"); 61. [String](http://www.google.com/search?hl=en&q=allinurl%3Adocs.oracle.com+javase+docs+api+string) str = console.nextLine(); 62. str = str.toLowerCase().trim(); 63. [String](http://www.google.com/search?hl=en&q=allinurl%3Adocs.oracle.com+javase+docs+api+string)[] facts = str.split(","); 64. handle\_facts(facts); 65. **if** (Objects.equals(str, "exit")) { 66. [System](http://www.google.com/search?hl=en&q=allinurl%3Adocs.oracle.com+javase+docs+api+system).out.println("Пока!"); 67. [System](http://www.google.com/search?hl=en&q=allinurl%3Adocs.oracle.com+javase+docs+api+system).exit(0); 68. } 69. } 70. } 72. **void** handle\_facts([String](http://www.google.com/search?hl=en&q=allinurl%3Adocs.oracle.com+javase+docs+api+string)[] facts) { 73. **for** ([String](http://www.google.com/search?hl=en&q=allinurl%3Adocs.oracle.com+javase+docs+api+string) fact : facts) { 74. **if** (fact.equals("help")) { 75. [System](http://www.google.com/search?hl=en&q=allinurl%3Adocs.oracle.com+javase+docs+api+system).out.println("Введите help, чтобы увидеть формат запросов"); 76. [System](http://www.google.com/search?hl=en&q=allinurl%3Adocs.oracle.com+javase+docs+api+system).out.println("Формат строки: <факт>, <факт> и так далее"); 77. [System](http://www.google.com/search?hl=en&q=allinurl%3Adocs.oracle.com+javase+docs+api+system).out.println("Примеры фактов, которые можно запросить у базы знаний:"); 78. [System](http://www.google.com/search?hl=en&q=allinurl%3Adocs.oracle.com+javase+docs+api+system).out.println("все бравлеры**\n**бравлеры <редкость> (например: бравлеры rare)**\n**кто сильнее <бравлер> <бравлер> (например: Кто сильнее max shelly)" + 79. "**\n** бой <бравлер> <бравлер>**\n** редкость <бравлер>**\n** синергия <бравлер>"); 80. } **else** { 81. **if** (fact.equals("все бравлеры")) { 82. data.getAllBrawlers(); 83. } **else** { 84. [String](http://www.google.com/search?hl=en&q=allinurl%3Adocs.oracle.com+javase+docs+api+string)[] words = fact.split(" "); 85. **if** (words[0].equals("бравлеры")) { 86. data.getAllBrawlersByRarity(words[1]); 87. } **else** **if** (words[0].equals("редкость")) { 88. data.rarityByBrawlerName(words[1]); 89. } **else** **if** (words[0].equals("синергия")) { 90. data.getBrawlerWithGoodTeam(words[1]); 91. } **else** **if** (words[0].equals("бой")) { 92. data.fight(words[1], words[2]); 93. } **else** **if** (words[0].equals("кто") && words[1].equals("сильнее")) { 94. data.morePower(words[2], words[3]); 95. } **else** { 96. [System](http://www.google.com/search?hl=en&q=allinurl%3Adocs.oracle.com+javase+docs+api+system).out.println("Факт: '" + fact + "' введён некорректно"); 97. } 98. } 99. } 101. } 102. } 103. } 104. **package** model; 106. **import** java.lang.reflect.Array; 107. **import** java.util.ArrayList; 108. **import** java.util.HashMap; 110. **public** **class** Data { 111. **private** ArrayList<Brawler> array = **new** ArrayList<>(); 112. **private** HashMap<[String](http://www.google.com/search?hl=en&q=allinurl%3Adocs.oracle.com+javase+docs+api+string), String> map\_good\_team = **new** HashMap<>(); 114. **public** ArrayList<Brawler> getArray() { 115. **return** array; 116. } 118. **public** **void** setArray(ArrayList<Brawler> array) { 119. **this**.array = array; 120. } 122. **public** HashMap<[String](http://www.google.com/search?hl=en&q=allinurl%3Adocs.oracle.com+javase+docs+api+string), String> getMap\_good\_team() { 123. **return** map\_good\_team; 124. } 126. **public** **void** setMap\_good\_team(HashMap<[String](http://www.google.com/search?hl=en&q=allinurl%3Adocs.oracle.com+javase+docs+api+string), String> map\_good\_team) { 127. **this**.map\_good\_team = map\_good\_team; 128. } 130. **public** **void** getAllBrawlers() { 131. **this**.array.**forEach**(x -> [System](http://www.google.com/search?hl=en&q=allinurl%3Adocs.oracle.com+javase+docs+api+system).out.printf("name: " + x.name + "**\n**" + 132. "rarity: " + x.rarity + "**\n**" + 133. "damage: " + x.damage + "**\n**" + 134. "health: " + x.health + "**\n\n**")); 135. } 137. **public** **void** getAllBrawlersByRarity([String](http://www.google.com/search?hl=en&q=allinurl%3Adocs.oracle.com+javase+docs+api+string) rarity) { 138. **this**.array.stream().filter(x -> x.rarity.equals(rarity)).**forEach**(x -> [System](http://www.google.com/search?hl=en&q=allinurl%3Adocs.oracle.com+javase+docs+api+system).out.println("name: " + x.name)); 139. } 141. **public** **void** getBrawlerWithGoodTeam([String](http://www.google.com/search?hl=en&q=allinurl%3Adocs.oracle.com+javase+docs+api+string) name) { 142. **if** (**this**.map\_good\_team.get(name) == **null**) { 143. [System](http://www.google.com/search?hl=en&q=allinurl%3Adocs.oracle.com+javase+docs+api+system).out.print("Для данного бравлера нет синергии либо такого бравлера вообще не существует**\n**"); 144. } **else** { 145. [System](http://www.google.com/search?hl=en&q=allinurl%3Adocs.oracle.com+javase+docs+api+system).out.println(**this**.map\_good\_team.get(name)); 146. } 147. } 149. **public** **void** morePower([String](http://www.google.com/search?hl=en&q=allinurl%3Adocs.oracle.com+javase+docs+api+string) name1, [String](http://www.google.com/search?hl=en&q=allinurl%3Adocs.oracle.com+javase+docs+api+string) name2) { 150. **int** damage1 = 0; 151. **int** damage2 = 0; 152. **for** (Brawler x : array) { 153. **if** (x.name.equals(name1)) { 154. damage1 = x.damage; 155. } 156. } 157. **for** (Brawler x : array) { 158. **if** (x.name.equals(name2)) { 159. damage2 = x.damage; 160. } 161. } 162. **if** (damage2 == 0) { 163. [System](http://www.google.com/search?hl=en&q=allinurl%3Adocs.oracle.com+javase+docs+api+system).out.println("Бравлер " + name2 + "не найден"); 164. **return** ; 165. } 166. **if** (damage1 == 0) { 167. [System](http://www.google.com/search?hl=en&q=allinurl%3Adocs.oracle.com+javase+docs+api+system).out.println("Бравлер " + name1 + "не найден"); 168. **return** ; 169. } 170. **if** (damage2 > damage1) { 171. [System](http://www.google.com/search?hl=en&q=allinurl%3Adocs.oracle.com+javase+docs+api+system).out.println("more power:" + name2); 172. } **else** **if** (damage1 > damage2) { 173. [System](http://www.google.com/search?hl=en&q=allinurl%3Adocs.oracle.com+javase+docs+api+system).out.println("more power:" + name1); 174. } **else** { 175. [System](http://www.google.com/search?hl=en&q=allinurl%3Adocs.oracle.com+javase+docs+api+system).out.println("brawlers have the same damage"); 176. } 177. } 179. **public** **void** rarityByBrawlerName([String](http://www.google.com/search?hl=en&q=allinurl%3Adocs.oracle.com+javase+docs+api+string) name) { 180. **this**.array.stream().filter(x -> x.name.equals(name)).**forEach**(x -> [System](http://www.google.com/search?hl=en&q=allinurl%3Adocs.oracle.com+javase+docs+api+system).out.println("rarity: " + x.rarity)); 181. } 183. **public** **void** fight([String](http://www.google.com/search?hl=en&q=allinurl%3Adocs.oracle.com+javase+docs+api+string) name1, [String](http://www.google.com/search?hl=en&q=allinurl%3Adocs.oracle.com+javase+docs+api+string) name2) { 184. **int** damage1 = 0; 185. **int** health1 = 0; 186. **int** damage2 = 0; 187. **int** health2 = 0; 188. **for** (Brawler x : array) { 189. **if** (x.name.equals(name1)) { 190. damage1 = x.damage; 191. health1 = x.health; 192. } 193. } 194. **for** (Brawler x : array) { 195. **if** (x.name.equals(name2)) { 196. damage2 = x.damage; 197. health2 = x.health; 198. } 199. } 200. **if** (damage2 == 0) { 201. [System](http://www.google.com/search?hl=en&q=allinurl%3Adocs.oracle.com+javase+docs+api+system).out.println("Бравлер " + name2 + "не найден"); 202. **return** ; 203. } 204. **if** (damage1 == 0) { 205. [System](http://www.google.com/search?hl=en&q=allinurl%3Adocs.oracle.com+javase+docs+api+system).out.println("Бравлер " + name1 + "не найден"); 206. **return** ; 207. } 208. **int** flag = 2; 209. **while** (**true**) { 210. **if** (health1 <= 0) { 211. flag = 1; 212. **break**; 213. } 214. **if** (health2 <= 0) { 215. flag = 0; 216. **break**; 217. } 218. health1 -= damage2; 219. health2 -= damage1; 220. } 221. **if** (flag == 1) { 222. [System](http://www.google.com/search?hl=en&q=allinurl%3Adocs.oracle.com+javase+docs+api+system).out.println("winner:" + name2); 223. } **else** **if** (flag == 0) { 224. [System](http://www.google.com/search?hl=en&q=allinurl%3Adocs.oracle.com+javase+docs+api+system).out.println("winner:" + name1); 225. } **else** { 226. [System](http://www.google.com/search?hl=en&q=allinurl%3Adocs.oracle.com+javase+docs+api+system).out.println("Haven't winner"); 227. } 228. } 229. } 231. **package** model; 233. **public** **class** Brawler { 234. [String](http://www.google.com/search?hl=en&q=allinurl%3Adocs.oracle.com+javase+docs+api+string) name; 235. [String](http://www.google.com/search?hl=en&q=allinurl%3Adocs.oracle.com+javase+docs+api+string) rarity; 236. **int** damage; 237. **int** health; 239. **public** Brawler([String](http://www.google.com/search?hl=en&q=allinurl%3Adocs.oracle.com+javase+docs+api+string) name, [String](http://www.google.com/search?hl=en&q=allinurl%3Adocs.oracle.com+javase+docs+api+string) rarity, **int** damage, **int** health) { 240. **this**.damage = damage; 241. **this**.health = health; 242. **this**.rarity = rarity; 243. **this**.name = name; 244. } 245. } |

**Оценка и интерпретация результатов**

Построив БЗ на основе предметной области по мобильной игре Brawl Stars, я считаю, что требования к реализации онтологий и БЗ соблюдены. БЗ и онтология связаны. БЗ и онтологии полны, удобочитаемы и точны. В целом по написанной рекомендательной системе можно вполне получить все необходимые знания по игре.

Запросы на Prolog:

|  |
| --- |
| 1. ?- more\_power(shelly, max, Result). 2. Result = shelly. 4. ?- damage\_by\_brawler(piper, X). 5. X = 3400. 7. ?- find\_brawlers\_by\_rarity(legendary, Brawlers). 8. Brawlers = [crow, spike, leon, amber, kordelius, surge, sandy]. 10. ?- fight(piper, bull). 11. piper наносит удар по bull на 3400 урона. Осталось здоровья у bull: 6600 12. bull наносит удар по piper на 880 урона. Осталось здоровья у piper: 3720 13. piper наносит удар по bull на 3400 урона. Осталось здоровья у bull: 3200 14. bull наносит удар по piper на 880 урона. Осталось здоровья у piper: 2840 15. piper наносит удар по bull на 3400 урона. Осталось здоровья у bull: -200 16. piper побеждает! 17. [true](http://pauillac.inria.fr/~deransar/prolog/bips.html). 19. ?- good\_team(tara, X). 20. X = sandy |

Онтограф (Семантическая сеть):

Изображение выглядит как текст, снимок экрана, программное обеспечение, линия

Автоматически созданное описание

Запросы на рекомендательной системе:

Изображение выглядит как текст, снимок экрана, Шрифт, меню

Автоматически созданное описание

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

Запросы на SPARQL Query:

PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>

PREFIX owl: <http://www.w3.org/2002/07/owl#>

PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>

PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>

SELECT ?subject ?predicate ?object

WHERE { ?subject ?predicate ?object }

PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>

PREFIX owl: <http://www.w3.org/2002/07/owl#>

PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>

PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>

SELECT ?subject ?object

WHERE { ?subject ?rdfs:subClassOf ?object }

PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>

PREFIX owl: <http://www.w3.org/2002/07/owl#>

PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>

PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>

SELECT ?subject ?object

WHERE { ?subject ?predicate owl:DatatypeProperty }

Исходя из полученных результатов систему ещё можно расширять до более глубоких знаний, но уже этого достаточно, чтобы получать всю информацию о персонажах, о том, кто сильнее, и о том какую сыгранность персонажи имеют между собой.

**Вывод**

**Преимущества системы ИИ на базе Prolog, баз знаний и онтологий:**

1. **Логический вывод и решение задач**:
   * **Prolog** предоставляет мощный механизм для логического вывода и решения задач. Он использует правила и факты для нахождения решений на основе логических связей. Это позволяет эффективно решать задачи, которые требуют поиска оптимальных решений или анализа сложных взаимосвязей.
2. **Гибкость и масштабируемость**:
   * Системы, основанные на базах знаний и онтологиях, легко расширяемы. Новые факты, правила и концепции могут быть добавлены без необходимости полного изменения системы. Это обеспечивает гибкость в обновлении базы знаний по мере появления новой информации.
3. **Автоматизация принятия решений**:
   * Разработанная система может автоматизировать процесс принятия решений, основанный на анализе данных и логическом выводе. Это особенно полезно в областях, где необходимо принимать сложные решения на основе большого количества параметров и ограничений.

**Потенциальные применения разработанной системы:**

1. **Экспертные системы**:
   * Система может быть использована для разработки экспертных систем в различных областях (например, диагностика заболеваний, юридическая поддержка, финансовые консультации). База знаний содержит правила и факты, которые позволяют системе делать точные выводы и предлагать решения на основе предоставленной информации.
2. **Автоматизация бизнес-процессов**:
   * В бизнесе такие системы могут применяться для автоматизации принятия решений, оптимизации логистических операций, оценки рисков или планирования производства. Логический вывод и структурированные знания позволяют находить оптимальные решения в условиях ограничений и неопределенности.
3. **Системы поддержки принятия решений (СППР)**:
   * Разработанная система может быть использована для разработки СППР в различных областях, таких как управление персоналом, планирование ресурсов, оптимизация процессов и т.д. СППР могут анализировать множество факторов и помогать пользователям принимать обоснованные решения на основе логических выводов.
4. **Обработка естественного языка**:
   * **Prolog** может применяться для задач обработки естественного языка (NLP), таких как синтаксический анализ, генерация ответов на вопросы и работа с грамматическими структурами. Система может использоваться в чат-ботах, системах поиска информации и инструментах автоматического перевода.