

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ
УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«Национальный исследовательский университет ИТМО»

ФАКУЛЬТЕТ ПРОГРАММНОЙ ИНЖЕНЕРИИ И КОМПЬЮТЕРНОЙ ТЕХНИКИ

ЛАБОРАТОРНАЯ РАБОТА №1

по дисциплине

«Системы искусственного интеллекта »

Выполнил работу:

Студент группы Р3318

Рамеев Тимур Ильгизович

Преподаватель:

Авдюшина Анна

Евгеньевна

Санкт-Петербург 2024

Содержание

Цель	2
Анализ требований	2
Основные концепции и инструменты	2
Реализация системы ИИ	2
Описание базы знаний	2
Запросы к базе знаний	4
Программная часть	4
База знаний	4
Запросы к базе знаний	7
Консольное приложение	7
Онтологический граф	11
Объектные свойства	12
Классы и их объекты	13
Запрос Protege	14
Вывод	16

Цель

Знакомство с базами знаний и онтологиями, языком Prolog и редактором онтологий Protege.

Анализ требований

Система должна предоставлять следующий функционал:

1. На основе предпочтений (любимого времени года, позиции и роли персонажа) советовать, ранжировать персонажей и предлагать наиболее подходящих для выбора
2. Выводить подробную информацию о заданном персонаже, а при некорректном вводе выводить список всех имеющихся персонажей
3. Выводить подробную информацию о заданной территории, а при некорректном вводе выводить список всех имеющихся территорий

Основные концепции и инструменты

В основе баз знаний лежат факты и правила. Факты представляют из себя верные отношения между сущностями, а правила - способы порождения новых фактов из имеющихся.

Для использования базы знаний необходимо задать цель - неизвестную переменную в предикате, которую база попытается подобрать так, чтобы предикат был верен. Язык Prolog позволяет описать базу знаний декларативно. В нем можно описать факты - верные предикаты с полными аргументами, а также задать аналоги функций - правила.

В проекте мы не будем работать с Prolog напрямую, а будем использовать интерпретатор `pyswip`.

Реализация системы ИИ

Описание базы знаний

База знаний, написанная на языке Prolog, содержит информацию о персонажах вселенной League Of Legends. В ней описаны класс персонажа, его роль на карте, место обитания (по лору), а также враждебные отношения между персонажами (по лору). Также база знаний содержит следующие правила:

1. Правило для поиска персонажей с одной территории
2. Правило для поиска персонажей с одинаковой ролью
3. Правило, ищущее всех топеров-стрелков
4. Правило для поиска войны между территориями
5. Правило для поиска врагов с одинаковой ролью на карте
6. Правило для поиска друзей (Территории с общим врагом без войны друг с другом)

Запросы к базе знаний

Помимо базы знаний, были написаны несколько запросов к ней, для демонстрации ее работы:

1. Посмотреть всех убийц
2. Посмотреть с какой территории персонаж Трэш
3. Посмотреть есть ли враги у Ка'Зикса
4. Посмотреть магов из Бандл-Сити, играющих на топе или на миде
5. Посмотреть всех врагов Ноксуса
6. Посмотреть всех воинов и убийц играющих против магов на одной позиции

Программная часть

База знаний

```
1  % Роль персонажа
2  tank(tarik).
3  tank(trash).
4  tank(braum).
5  wizard(zerat).
6  wizard(ari).
7  wizard(sweyn).
8  wizard(reykan).
9  wizard(veigar).
10 wizard(luxe).
11 warrior(vai).
12 warrior(viego).
13 warrior(darius).
14 warrior(saylas).
15 warrior(warwick).
16 warrior(garen).
17 killer(khaZix).
18 killer(rengar).
19 killer(pyke).
20 killer(zed).
21 killer(ekko).
22 killer(katarina).
23 shooter(timo).
24 shooter(ashе).
```

```

25 shooter(senna).
26 shooter(lucian).
27 shooter(dreivan).
28 shooter(jinx).
29
30 % Родина персонажа ( по лору )
31 homeland(shadowyIslands, lucian).
32 homeland(shadowyIslands, senna).
33 homeland(shadowyIslands, viego).
34 homeland(shadowyIslands, trash).
35 homeland(gapingHole, khaZix).
36 homeland(bildgwater, pyke).
37 homeland(ionia, zed).
38 homeland(ionia, rengar).
39 homeland(ionia, reykan).
40 homeland(nocsus, sweyn).
41 homeland(nocsus, dreivan).
42 homeland(nocsus, darius).
43 homeland(nocsus, katarina).
44 homeland(zaun, jinx).
45 homeland(zaun, vai).
46 homeland(zaun, ekko).
47 homeland(zaun, warwick).
48 homeland(demasiya, luxe).
49 homeland(demasiya, garen).
50 homeland(freljord, saylas).
51 homeland(freljord, ashe).
52 homeland(freljord, braum).
53 homeland(targon, tarik).
54 homeland(bandlCity, veigar).
55 homeland(bandlCity, timo).
56 homeland(bandlCity, ari).
57 homeland(shurima, zerat).
58
59
60 % Место отыгрыша на карте (Некорректно немного получилось)
61 role(carry, lucian).
62 role(support, trash).
63 role(support, senna).
64 role(jungle, viego).
65 role(jungle, khaZix).

```

```

66 role(support, pyke).
67 role(mid, zed).
68 role(jungle, rengar).
69 role(support, reykan).
70 role(support, sweyn).
71 role(carry, dreivan).
72 role(top, darius).
73 role(carry, jinx).
74 role(jungle, vai).
75 role(jungle, ekko).
76 role(jungle, warwick).
77 role(mid, luxe).
78 role(top, garen).
79 role(mid, saylas).
80 role(carry, ashe).
81 role(support, braum).
82 role(support, tarik).
83 role(mid, veigar).
84 role(top, timo).
85 role(mid, ari).
86 role(mid, zerat).
87 role(mid, katarina).
88
89 % Враги (по лопу )
90 enemies(khaZix, rengar).
91 enemies(saylas, garen).
92 enemies(garen, darius).
93 enemies(katarina, ari).
94 enemies(timo, pyke).
95 enemies(senna, trash).
96 enemies(senna, viego).
97 enemies(lucian, trash).
98 enemies(lucian, viego).
99 enemies(jinx, vai).
100
101 % Персонажи с одной территории
102 countrymans(X, Y) :- homeland(Z, X), homeland(H, Y), Z = H, X \= Y.
103
104 % Персонажи с одинаковой ролью
105 sameRoles(X, Y) :- role(Z, X), role(H, Y), Z = H, X \= Y.
106

```

```

107 % Топеры-стрелки
108 topShooters(X) :- role(top, X), shooter(X).
109
110 % Войны между территориями
111 wars(X, Y) :- homeland(X, Z), homeland(Y, H), (enemies(Z, H); enemies(H, Z))
112
113 % Враги с одинаковой позицией в игре
114 sameEnemiesByRole(X, Y) :- role(Z, X), role(H, Y), (enemies(X, Y); enemies(Y
115
116 % Друзья (Территории с общим врагом без войны друг с другом)
117 friends(X, Y) :- wars(X, Z), wars(Y, Z), not(wars(X, Y)), X \= Y.

```

Запросы к базе знаний

```

1 % посмотреть всех убийц
2 killer(X).
3
4 % посмотреть с какой территории Треш
5 homeland(X, trash).
6
7 % посмотреть, есть ли враги у казика
8 enemies(khaZix, _); enemies(_, khaZix).
9
10 % посмотреть магов из Бандл-Сити, играющих на топе или на миде
11 wizard(X), homeland(bandlCity, X), (role(top, X); role(mid, X)).
12
13 % посмотреть всех врагов Ноксуса
14 {Z}/(wars(nocsus, Y), homeland(Y, Z)).
15
16 % посмотреть всех воинов и убийц играющих против магов на одной линии
17 (killer(X); warrior(X)), wizard(Y), role(Role, X), role(Role, Y).

```

Консольное приложение

Написано на языке Python

```

1 from pyswip import Prolog
2
3 def start_dialog():
4     while True:

```



```

5     print("Привет! Меня зовут БОБ! Я - база знаний по игре League of Leg
6     print("a) Подобрать персонажа")
7     print("b) Узнать информацию о персонаже")
8     print("c) Посмотреть друзей и врагов выбранной территории")
9     print("d) Выход")
10    option = input().strip()
11    if option == "a":
12        find_character_manager()
13    elif option == "b":
14        find_info_character_manager()
15    elif option == "c":
16        find_info_territory_manager()
17    elif option == "d":
18        break
19    else:
20        print("Похоже я вас не понимаю, пожалуйста попробуйте еще раз")
21
22
23    def find_info_territory_manager():
24        print("Вы выбрали опцию \"Посмотреть друзей и врагов выбранной территории\"")
25        name_of_territory = input()
26        wars = [x['X'] for x in list(prolog.query(f'wars(X, {name_of_territory})'))
27        friends = [x['X'] for x in list(prolog.query(f'friends(X, {name_of_territory})'))
28        valid_name = list(prolog.query(f'homeland({name_of_territory}, X)'))
29        if len(valid_name) > 0:
30            print("Итак, я нашел некоторую информацию об этой территории:")
31            print(f"Наименование территории: {name_of_territory}")
32            print(f"Враждебные территории: {wars}")
33            print(f"Союзные территории: {friends}")
34        else:
35            print("Извините я не нашел такой территории, вот, на всякий случай, ")
36            all_lands = list(set(x['X'] for x in list(prolog.query("homeland(X, Y)"))))
37            for i in range(1, len(all_lands) + 1):
38                print(i, " : ", all_lands[i - 1])
39
40    def find_info_character_manager():
41        print("Вы выбрали опцию \"Узнать информацию о персонаже\". Отлично! Введите имя персонажа")
42        name_of_character = input()
43        homeland = list(prolog.query(f'homeland(X, {name_of_character})'))
44        role = list(prolog.query(f'role(X, {name_of_character})'))
45        position = list(prolog.query(f'position(X, {name_of_character})'))

```

```

46 enemies = list(prolog.query(f'enemies(X, {name_of_character}); enemies({
47 countrymans = list(prolog.query(f'countrymans(X, {name_of_character}'))
48 if len(homeland) == 1:
49     print("Итак, я нашел некоторую информацию об этом парнише:")
50     print(f"Имя персонажа: {name_of_character}")
51     print(f"Родина персонажа: {homeland[0]['X']}")
52     print(f"Роль персонажа: {role[0]['X']}")
53     print(f"Позиция персонажа: {position[0]['X']}")
54     if len(enemies) > 0:
55         print("Враги персонажа: ", [x['X'] for x in enemies])
56     else:
57         print("У персонажа отсутствуют враги")
58     if len(countrymans) > 0:
59         print("Соотечественники персонажа: ", [x['X'] for x in countrymans])
60     else:
61         print("У персонажа отсутствуют соотечественники")
62 else:
63     print("Извините, я не нашел такого персонажа, вот, на всякий случай,
64     all_chars = [x['X'] for x in list(prolog.query("character(X)"))]
65     for i in range(1, len(all_chars) + 1):
66         print(i, " : ", all_chars[i - 1])
67
68
69 def find_character_manager():
70     while True:
71         try:
72             print("Вы выбрали опцию \"Подобрать персонажа\". Отлично! Сейчас
73             result_list = homeland_choise() + role_choise() + position_choise()
74             result_dict = {1 : set(), 2 : set(), 3 : set()}
75             for item in result_list:
76                 result_dict[result_list.count(item)].add(item)
77
78             print("Итак, вот персонажи, которых я для Вас нашел:")
79             print("Наилучшая синергия:", result_dict[3])
80             print("Вы неплохо поладите:", result_dict[2])
81             print("Будет тяжело освоиться:", result_dict[1])
82             print("Остальных персонажей крайне не рекомендую. Удачи, боец!")
83             break
84         except Exception as unrecognized_input:
85             print("Похоже я вас не понимаю, пожалуйста попробуйте еще раз")
86             continue

```

```

87
88
89 def homeland_choise():
90     print("Какое время года Вам нравится больше всего?")
91     print("a) Зима")
92     print("b) Весна")
93     print("c) Лето")
94     print("d) Осень")
95     season = input().strip()
96     if season == "a":
97         season = "winter"
98     elif season == "b":
99         season = "spring"
100    elif season == "c":
101        season = "summer"
102    elif season == "d":
103        season = "autumn"
104    else:
105        raise Exception
106    return [x['Y'] for x in list(prolog.query(f'homeland(X, Y), season({season}')))]
107
108 def role_choise():
109     print("Какая роль вам нравится больше всего?")
110     print("a) Убийца")
111     print("b) Танк")
112     print("c) Маг")
113     print("d) Стрелок")
114     print("e) Воин")
115     role = input().strip()
116     if role == "a":
117         role = "killer"
118     elif role == "b":
119         role = "tank"
120     elif role == "c":
121         role = "wizard"
122     elif role == "d":
123         role = "shooter"
124     elif role == "e":
125         role = "warrior"
126     else:
127         raise Exception

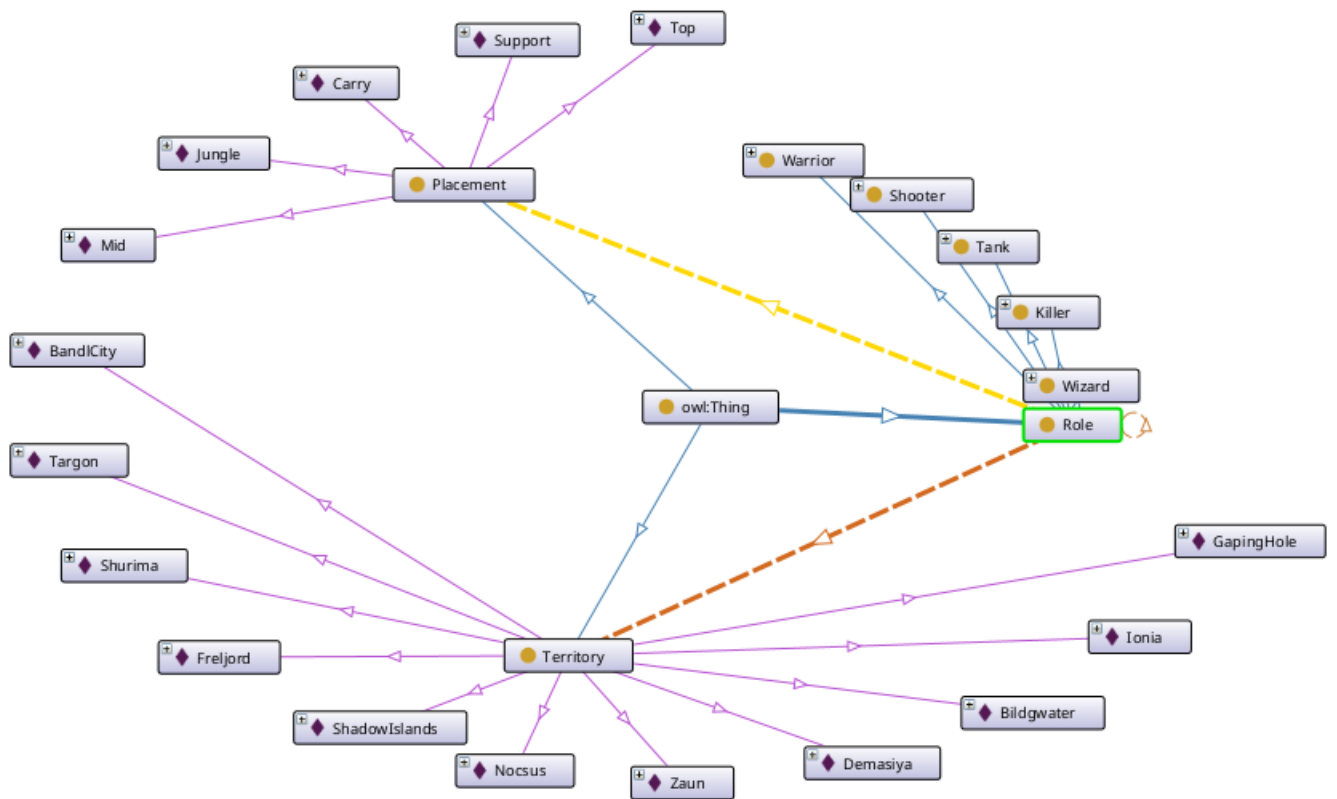
```

```

128     return [x['Y'] for x in list(prolog.query(f'role({role}, Y)'))]
129
130 def position_choise():
131     print("На какой позиции на карте вы бы хотели отыгрывать?")
132     print("a) Топ")
133     print("b) Керри")
134     print("c) Саппорт")
135     print("d) Лес")
136     print("e) Мид")
137     position = input().strip()
138     if position == "a":
139         position = "top"
140     elif position == "b":
141         position = "carry"
142     elif position == "c":
143         position = "support"
144     elif position == "d":
145         position = "jungle"
146     elif position == "e":
147         position = "mid"
148     else:
149         raise Exception
150     return [x['Y'] for x in list(prolog.query(f'position({position}, Y)'))]
151
152 prolog = Prolog()
153 prolog.consult("Lab1/knowledgeBase.pl")
154 start_dialog()

```

Онтологический граф



Объектные свойства

Object property hierarchy: toHaveEnemies

Annotations Usage

Annotations: toHaveEnemies

Annotations +

Characteristics: toHave

- ☐ Functional
- ☐ Inverse functional
- ☐ Transitive
- ☒ Symmetric
- ☐ Asymmetric
- ☐ Reflexive
- ☒ Irreflexive

Git: main

The screenshot shows a web-based OWL editor interface. The main panel displays the 'Object property hierarchy: toHaveEnemies'. On the left, a tree view shows the hierarchy: 'owl:topObjectProperty' is the parent, and 'toHaveEnemies', 'toHaveHomeland', and 'toHavePlacement' are its children. The 'toHaveEnemies' property is highlighted with a blue border. On the right, there are two panels. The top panel, titled 'Annotations: toHaveEnemies', is currently empty. The bottom panel, titled 'Characteristics: toHave', contains a list of checkboxes for property characteristics: 'Functional', 'Inverse functional', 'Transitive', 'Symmetric' (checked), 'Asymmetric', 'Reflexive', and 'Irreflexive' (checked). At the bottom left of the editor, a status bar indicates 'Git: main'.

Классы и их объекты

The screenshot displays a web application interface for managing classes and individuals. The top-left pane shows a class hierarchy starting with `owl:Thing`, which includes `Placement`, `Role`, `Killer`, `Shooter`, `Tank`, `Warrior` (highlighted), `Wizard`, `Zaun_homeland`, and `Territory`. `Zaun_wars` is listed under `Territory`. The top-right pane shows the `Warrior` class with a list of individuals: `Ari`, `Ashe`, `BandICity`, `Bildgwater`, `Braum`, `Carry`, `Dem`, `GapingHole`, `Garen`, `Ionia`, `Jinx`, `Jungle`, `Katarina`, `Kha`, `Rengar`, `Reykan`, `Saylas`, `Senna`, `ShadowIslands`, `Shur`, `Timo`, `Top`, `Trash`, `Vai`, `Veigar`, `Viego`, `Warwick`, and `Zaun`. The bottom-left pane shows the `Warrior` class with a list of individuals: `Darius` (highlighted), `Garen`, `Saylas`, `Vai`, `Viego`, and `Warwick`. The bottom-right pane shows the `Warrior` class with a list of individuals: `Darius` (highlighted), `Garen`, `Saylas`, `Vai`, `Viego`, and `Warwick`. The bottom status bar indicates the Git branch is `main`.

owl:Thing

- Placement
- Role
 - Killer
 - Shooter
 - Tank
 - Warrior
 - Wizard
- Zaun_homeland
- Territory
 - Zaun_wars

Types +

- Warrior

Same Individual As +

Different Individuals +

- Ari, Ashe, BandICity, Bildgwater, Braum, Carry, Dem, GapingHole, Garen, Ionia, Jinx, Jungle, Katarina, Kha, Rengar, Reykan, Saylas, Senna, ShadowIslands, Shur, Timo, Top, Trash, Vai, Veigar, Viego, Warwick, Zaun, Z

Individuals Individuals (Inferred)


Direct instances: Darius

For: Warrior

- Darius
- Garen
- Saylas
- Vai
- Viego
- Warwick

Git: main

Запрос Protege

Equivalent To 

● **Territory** and (**inverse** (toHaveHomeland) **some**
(Role and (not (toHaveHomeland **value** Nocsus)) and (toHaveEnemies **some**
(Role and (toHaveHomeland **value** Nocsus))))))

Вывод

Мы познакомились с основными концепциями баз знаний и онтологий, освоили язык Prolog и редактор онтологий Protege. Базы знаний на Prolog позволяют удобно осуществлять резолюцию целей по имеющимся фактам, а система Protege может выявлять несоответствия и "скрытую информацию" в разработанных онтологиях.