Университет ИТМО
Факультет программной инженерии и компьютерной техники

Системы искусственного интеллекта. Отчет по модулю 1.

Группа : P33151
Студент: Кортыш Андрей Олегович
Тема : Синергия героев в игре Dota 2

2023

Содержание

1 Введение **1**
1.1 Цели . . . 1
1.2 Значимость . . . 1

2 Анализ требований **1**
2.1 Требования к базе знаний . . . 1
2.2 Требования к системе поддержки принятия решений . . . 2

3 Изучение основных концепций и инструментов **2**
3.1 Обзор основных концепций баз знаний и онтологий . . . 2
3.2 Изучение Prolog . . . 2
3.3 Инструменты и библиотеки, подходящие для работы с базами знаний и онтологиями на Prolog . . . 2

4 Реализация системы поддержки принятия решений при помощи базы знаний **2**
4.1 Ссылка на исходные коды . . . 2
4.2 База знаний в Prolog . . . 3
4.3 Тестирование и отладка системы, обеспечение ее функциональности и эффективности 3
4.4 Пример вывода программы . . . 4

5 Оценка и интерпретация результатов **4**
5.1 Примеры запросов к базе знаний и онтологии . . . 4
5.2 Оценка системы и дальнейшего равития . . . 5

6 Заключение **5**

1 Введение

1.1 Цели

Целью лабораторных работ этого модуля была разработка базы знаний на языке Prolog и онтологии в Protege. А после этого разработка пользовательского интерфейса для упрощения взаимодействия пользователя с ними.

1.2 Значимость

Значимость выбранной мной темы заключается в помощи начинающим игрокам, так как Dota 2 очень сложная и комплексная игра с множеством героев и механик, которые различным образом взаимодействуют друг с другом. Таким образом, разработанная мной система поддержки принятия решений поможет этим игрокам быстрее освоиться и разобраться в игре и побеждать в большем количестве игр.

2 Анализ требований

2.1 Требования к базе знаний

1. База знаний должна быть реализована на языке Prolog.
2. База знаний должна содержать информацию о героях Dota 2 и их способностях.
3. База знаний должна содержать в себе правила для определения типа героя.
4. База знаний должна содержать в себе правила для определения синергии между героями.

2.2 Требования к системе поддержки принятия решений

1. Система должна иметь возможность обращения к сформированной ранее базе знаний.
2. Система должна предоставлять пользователю возможность модификации запросов.
3. Система должна корректно реагировать на ошибочный ввод пользователя.
4. Система должна предоставлять пользователю варианты выбора для пропусков, где выбор существенно ограничен.

3 Изучение основных концепций и инструментов

3.1 Обзор основных концепций баз знаний и онтологий

База знаний - это база данных, содержащая в себе знания о некоторой предметной области и человеческом опыте.
Онтология представляет собой иерхахический способ представления понятий и связей между ними. При добавлении в онтологию информацию о конкретных объектов часто называют базой знаний. Основным отличием баз знаний от привычных всем баз данных яляется возможность при вводе в систему правил вывода делать автоматические умозаключения на основе имеющихся фактов и понятий и тем самым производить семантическую обработку информациии.

Основными элементами онтологий являются:

1. Экземпляры (instances) - это объекты, основные и, при этом, нижнеуровневые элементы онтологии.
2. Понятия (classes) - абстрактные коллекции или наборы объектов.
3. Атрибуты - объекты могут иметь атрибуты, которые содержат информацию об объектах и привязаны к ним.
4. Отношения - играют важную связь в связи объектов онтологии между собой. Обычно отношения - это атрибуты значениями, которых является другие объекты.

3.2 Изучение Prolog

Prolog - это декларативный язык программирования. Программы на данном языке представляют собой набор фактов и правил. При выполнении запроса к Prolog выполняется обращение в базу знаний, на которые в простом случае (отсутсвии переменных в запросе) выдается булево значение: true или false, в более сложном случае система выведет конкретные данные, для которых значение соотвествующих правил истинно.

3.3 Инструменты и библиотеки, подходящие для работы с базами знаний и онтологиями на Prolog

Для установки Prolog можно воспользоваться: `https://www.swi-prolog.org/download/stable`.
Для установки Protege: `https://protegeproject.github.io/protege/installation/linux/`.
Библиотека для вызова Prolog из Python: `https://github.com/yuce/pyswip`.

4 Реализация системы поддержки принятия решений при помощи базы знаний

4.1 Ссылка на исходные коды

`https://github.com/akorton/AI_systems`

4.2 База знаний в Prolog

Пример кода:

```
% hero(hero_name)
hero("AXE").
hero("BANE").
hero("BATRIDER").
hero("BEASTMASTER").
hero("CRYSTAL MAIDEN").
hero("DARK SEER").
hero("DARK WILLOW").
hero("EARTH SPIRIT").
hero("EARTHSHAKER").
hero("ELDER TITAN").
hero("ENCHANTRESS").
hero("ENIGMA").
hero("FACELESS VOID").
hero("GYROCOPTER").
hero("HOODWINK").
hero("HUSKAR").
hero("INVOKER").
hero("JAKIRO").
hero("JUGGERNAUT").
hero("LEGION COMMANDER").
hero("LINA").
hero("LION").
hero("MAGNUS").
hero("MONKEY KING").
```

4.3 Тестирование и отладка системы, обеспечение ее функциональности и эффективности

Для тестирования в Prolog были написаны автотесты с цветовой подсветкой (красная для упавших тестов, зеленая для прошедших):

```
    print_test(Test, Expected_result) :- ((Test, Expected_result); (\+ Test, \+
    Expected_result)),
    ansi_format([bold, fg(green)], 'Test: ~w passed\n', [Test]).
print_test(Test, Expected_result) :- ansi_format([bold, fg(red)], 'Test: ~w failed\n', [
    Test]).
print_test(Test, Expected_result, Custom_test_name) :- ((Test, Expected_result); (\+
    Test, \+ Expected_result)),
    ansi_format([bold, fg(green)], 'Test: ~w passed\n', [Custom_test_name]).
print_test(Test, Expected_result, Custom_test_name) :- ansi_format([bold, fg(red)], '
    Test: ~w failed\n', [Custom_test_name]).

?- writeln("").
?- writeln("Synergy/2 tests.").
?- print_test(heroes_synergy("AXE", "INVOKER"), true). % stun_aoe, dmg_aoe
?- print_test(heroes_synergy("INVOKER", "AXE"), true). % dmg_aoe, stun_aoe
?- print_test(heroes_synergy("BANE", "INVOKER"), true). % stun_single, dmg_single
?- print_test(heroes_synergy("INVOKER", "BANE"), true). % dmg_single, stun_single
?- print_test(heroes_synergy("BATRIDER", "CRYSTAL MAIDEN"), false). % dmg_single,
    dmg_aoe
?- print_test(heroes_synergy("EARTH SPIRIT", "ENCHANTRESS"), false). % dmg_aoe,
    dmg_single
?- print_test(heroes_synergy("HOODWINK", "LINA"), false). % dmg_single, dmg_single
?- print_test(heroes_synergy("JUGGERNAUT", "MONKEY KING"), false). % dmg_aoe, dmg_aoe
?- print_test(heroes_synergy("BEASTMASTER", "JAKIRO"), false). % stun_single, stun_aoe
?- print_test(heroes_synergy("DARK SEER", "BANE"), false). % stun_aoe, stun_single
?- print_test(heroes_synergy("BATRIDER", "BEASTMASTER"), false). % stun_single,
    stun_single
?- print_test(heroes_synergy("DARK WILLOW", "FACELESS VOID"), false). % stun_aoe,
    stun_aoe
?- print_test(heroes_synergy("EARTHSHAKER", "EARTHSHAKER"), false). % Can't be synergy
    with itself
?- print_test(heroes_synergy("HUSKAR", "HUSKAR"), false). % Can't be synergy with itself
?- writeln("").
?- writeln("Synergy/3 tests.").
```

```
?- print_test(heroes_synergy("ENIGMA", "DARK SEER", "JAKIRO"), true). % Some synergy/3 tests
?- print_test(heroes_synergy("ELDER TITAN", "MAGNUS", "CRYSTAL MAIDEN"), true). % Some synergy/3 tests
?- print_test(heroes_synergy("EARTHSHAKER", "ELDER TITAN", "GYROCOPTER"), true). % Some synergy/3 tests
?- print_test(heroes_synergy("DARK WILLOW", "BANE", "HUSKAR"), true). % Some synergy/3 tests
% Check for synergy with specific hero which should have stun, dmg ability or both
?- writeln("").
?- writeln("Complex synergy tests tests.").
?- print_test((findall(H, (hero(H), heroes_synergy(H, "INVOKER"), universal_hero(H)), L), member("HUSKAR", L)), true, "Universal hero synergy with INVOKER").
?- print_test((findall(H, (hero(H), heroes_synergy(H, "GYROCOPTER"), universal_hero(H)), L), member("ENIGMA", L)), true, "Universal hero synergy with GYROCOPTER").
?- print_test((findall(H, (hero(H), heroes_synergy(H, "BANE"), hero_with_stun(H)), L), member("DARK WILLOW", L)), true, "Hero with stun synergy with BANE").
?- print_test((findall(H, (hero(H), heroes_synergy(H, "JUGGERNAUT"), hero_with_stun(H)), L), member("BATRIDER", L)), true, "Hero with stun synergy with JUGGERNAUT").
?- print_test((findall(H, (hero(H), heroes_synergy(H, "LION"), hero_with_dmg(H)), L), member("HUSKAR", L)), true, "Hero with damage synergy with LION").
?- print_test((findall(H, (hero(H), heroes_synergy(H, "MONKEY KING"), hero_with_dmg(H)), L), member("JAKIRO", L)), true, "Hero with damage synergy with MONKEY KING").
?- writeln("").
```

Тестирование системы принятия решений на языке Python производилось вручную.

4.4 Пример вывода программы

```
1. Какой герой хорош в связке с _
2. Какой герой хорош в связке с _ и _
3. Я хочу пикнуть героя _ в связке с _
4. Я хочу пикнуть героя _ в связке с _ и _
Введите номер шаблона для поиска в базе знаний или 0 для выхода:
4
Я хочу пикнуть героя $1 в связке с $2 и $3
Возможные варианты для данного пропуска:
со станом
с уроном
универсального
Ввод для пропуска $1: со станом
Ввод для пропуска $2: EARTHSHAKER
Ввод для пропуска $3: ENIGMA
Нашлось 7 ответов.
Сколько из них вывести: 7
ELDER TITAN
FACELESS VOID
DARK SEER
AXE
JAKIRO
DARK WILLOW
MAGNUS
Хорошего дня!
```

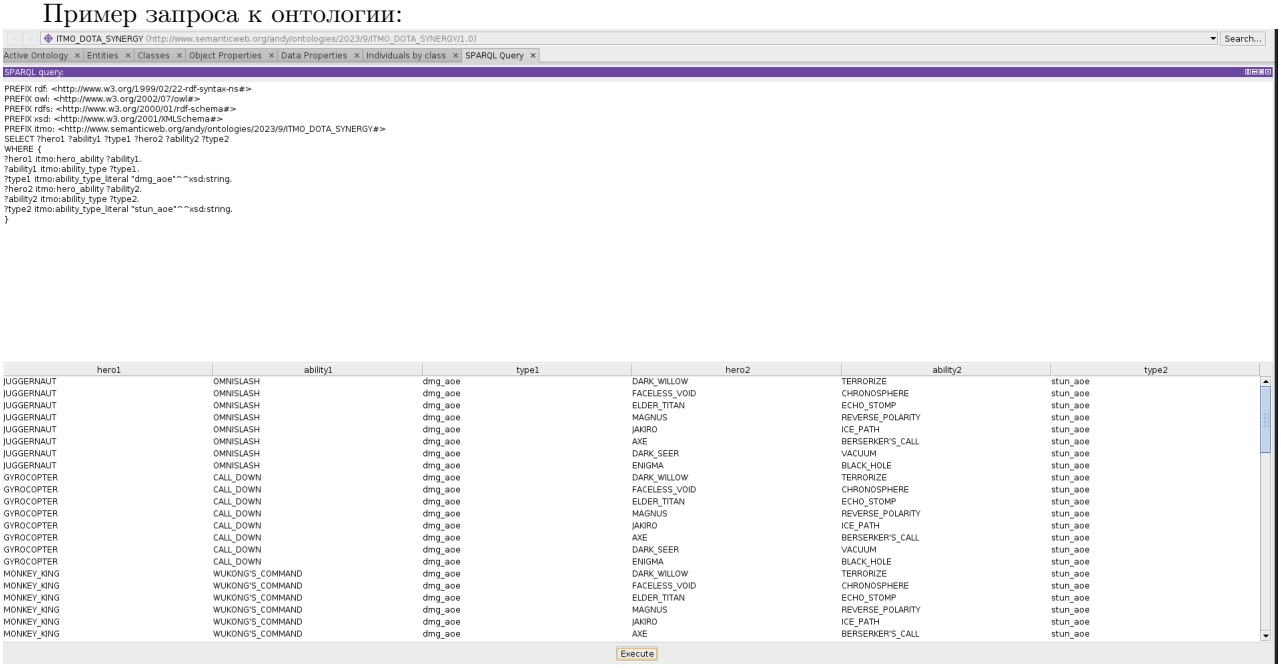
5 Оценка и интерпретация результатов

5.1 Примеры запросов к базе знаний и онтологии

Пример запроса к базе знаний:

```
?- findall(H, (heroes:hero(H), heroes_synergy(H, "INVOKER"), abilities:universal_hero(H)), L).
L = ["DARK WILLOW", "EARTHSHAKER", "ELDER TITAN", "ENIGMA", "HUSKAR", "JAKIRO"].
```

Пример запроса к онтологии:



5.2 Оценка системы и дальнейшего равития

Система соотвествует заявленным требованиям, так как в ней реализованы все заявленные возможности.
Пользователю системы предоставляется достаточно широко кастомизируемый ввод запросов 4-ех типов.
В базе знаний заложены сведения о более чем 25 героях и более чем 25 способностях и их типах, а также связи между ними.

В дальнейшем, конечно, можно добавить в базу знаний информацию обо всех 124 героях и их способностях. Кроме этого, на данный момент система достаточно примитивна и не учитывает такие особенности, как основной атрибут героя, его показатели силы, интеллекта и ловкости, их приросты и т.д. Все это и многое другое можно добавить в базу знаний и адаптировать правила для определения синергии, чтобы они учитывали данные параметры.

6 Заключение

Разработанная система поддержки принятия решений хоть и очень ограничена в своих возможностях, но, тем не менее, может быть действительно полезна начинающим игрокам. А мне очень понравилось разбираться в языке Prolog, так как он, по крайней мере для меня, был уникальным опытом и представлял собой совершенно новую область программирования, о которой я никогда не слышал.