

Университет ИТМО
Факультет программной инженерии и компьютерной техники

**Системы искусственного интеллекта.
Лабораторные работы 1-3.
Базы знаний и онтологии**

Группа : Р33131
Студент: Смирнов Виктор Игоревич
Тема : Игра Terraria

Ключевые слова

База знаний, Онтология.

Содержание

| | |
|---|----------|
| 1 Введение | 1 |
| 1.1 Цели | 1 |
| 1.2 Значимость | 1 |
| 2 Анализ требований | 1 |
| 2.1 Требования к фронтэнду | 1 |
| 2.2 Требования к базе знаний | 2 |
| 3 Изучение основных концепций и инструментов | 2 |
| 3.1 Обзор основных концепций баз знаний и онтологий | 2 |
| 3.2 Prolog и его возможностей для разработки систем искусственного интеллекта | 2 |
| 3.3 Инструменты и библиотеки, подходящие для работы с базами знаний и онтологиями на Prolog | 2 |
| 4 Создание правил и логики вывода для принятия решений на основе базы знаний и онтологии | 2 |
| 4.1 Создание правил и логики вывода для принятия решений на основе базы знаний и онтологии | 2 |
| 4.2 Онтология в Protege | 6 |
| 4.3 Тестирование и отладка системы, обеспечение ее функциональности и эффективности | 6 |
| 4.4 Пример вывода программы | 7 |
| 5 Оценка и интерпретация результатов | 7 |
| 6 Заключение | 7 |

1 Введение

1.1 Цели

Целью лабораторных работ первого модуля была разработка базы знаний на базе языка Prolog, а также фронтэнда к ней на Python для взаимодействия пользователя с системой. Прежде всего это был просто образовательный проект с целью ознакомления с онтологиями.

1.2 Значимость

Значимость заключается в полученных мною знаний и создании автоматического чат-бота для получения персональных рекомендаций по игре Terraria.

2 Анализ требований

2.1 Требования к фронтэнду

1. Фронтэнд должен обрабатывать запросы пользователя и выдавать на них ответы, соответствующие состоянию базы знаний.
2. Фронтэнд должен быть устойчив к некорректному пользовательскому вводу.
3. Фронтэнд должен быть реализован на языке Python.
4. Фронтэнд должен иметь CLI интерфейс.
5. Фронтэнд должен инициировать взаимодействие с базой знаний.

2.2 Требования к базе знаний

1. База знаний должна быть реализована на базе системы Prolog.
2. База знаний должна содержать информацию об игре Tetraia.
3. База знаний должна содержать факты и правила различной сложности.

3 Изучение основных концепций и инструментов

3.1 Обзор основных концепций баз знаний и онтологий

По определению из википедии база знаний - база данных, содержащая правила вывода и информацию о знаниях в некоторой предметной области.

Онтология – иерархический способ представления в базе знаний набора понятий и их связей.

Важно, что онтология сама по себе содержит только абстрактные понятия и связи. Она становится базой знаний, только когда там появляются конкретные объекты.

Базы знаний способны делать автоматические умозаключения об уже имеющихся или вновь вводимых фактах и тем самым производить как бы осмысленную обработку информации.

База знаний содержит правила вывода – логические конструкции, определяющие отношения между фактами и позволяющие делать выводы.

База знаний обрабатывает запросы к себе, выдавая информацию, удовлетворяющую заданным в запросе условиям.

Онтологии состоят из

1. экземпляров - конкретных или абстрактных объектов,
2. понятий - абстрактных наборов объектов,
3. атрибутов - некоторых значений, связанных с конкретным экземпляром,
4. отношений - зависимостей между объектами онтологии.

Онтология позволяет формализовать факты и правила и обрабатывать их автоматически, например, в системах искусственного интеллекта или для анализа данных.

Онтология задает смысл данных, а база знаний их хранит и предоставляет доступ.

3.2 Prolog и его возможностей для разработки систем искусственного интеллекта

Prolog (Programming in Logic) - это декларативный язык программирования, который используется для решения задач искусственного интеллекта и логического программирования. Prolog основан на логике первого порядка и предназначен для решения задач, в которых логические отношения и правила играют ключевую роль.

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

Инструмент для Базы знаний - <https://swish.swi-prolog.org/>.

Инструмент для Онтологий - <https://protege.stanford.edu/>.

Инструмент для взаимодействия с базой знаний - <https://github.com/yuce/pyswip/>.

4 Создание правил и логики вывода для принятия решений на основе базы знаний и онтологии

4.1 Создание правил и логики вывода для принятия решений на основе базы знаний и онтологии

Была составлена база знаний на языке Prolog.

```

1 :- module(achievements, [achived/1]).
2
3 :- use_module(ore).
4 :- use_module(pickaxe).
5 :- use_module(available).
6 :- use_module(reachable).
7
8 achived('Can break all found existing ores') :-
9     forall(
10         found(existing(ore(Ore))),
11         available(ore(Ore))
12     ).
13
14 achived('Theoretically can craft both Nightmare and Deathbringer') :-
15     reachable(pickaxe('Nightmare')),
16     reachable(pickaxe('Deathbringer')).
17
18 achived('Is ready to hardmode') :-
19     available(pickaxe('Molten')).

```

Листинг 1: Достижения игрока

```

1 :- module(available, [available/1]).
2
3 :- use_module(progress).
4 :- use_module(requirement).
5
6 available(Item) :-
7     Item, found(Item).
8
9 available(ore(Ore)) :-
10     found(existing(ore(Ore))),
11     ore_power(ore(Ore), OrePower),
12     findall(Pickaxe, (
13         pickaxe(Pickaxe),
14         pickaxe_power(pickaxe(Pickaxe), PickaxePower),
15         (PickaxePower < OrePower -> false; true),
16         found(pickaxe(Pickaxe))
17     ), Pickaxes),
18     member(_, Pickaxes).
19
20 available(Item) :-
21     Item,
22     findall(R, requirement(Item, R), Requirements),
23     member(_, Requirements),
24     forall(member(R, Requirements), found(R)).

```

Листинг 2: Доступность предмета игроку в один шаг

```

1 :- module(reachable, [reachable/1]).
2
3 :- use_module(requirement).
4 :- use_module(available).
5
6 :- use_module(pickaxe).
7
8 reachable(Item) :-
9     Item, available(Item).
10
11 reachable(pickaxe_with_power(TargetPower)) :-
12     findall(Pickaxe, (
13         pickaxe(Pickaxe),
14         pickaxe_power(pickaxe(Pickaxe), PickaxePower),
15         (PickaxePower < TargetPower -> false; true)
16     ), Pickaxes),
17     member(FirstPickaxe, Pickaxes),
18     reachable(pickaxe(FirstPickaxe)).
19
20 reachable(Item) :-
21     Item,
22     findall(R, requirement(Item, R), Requirements),
23     member(_, Requirements),

```

```
24 forall(member(R, Requirements), reachable(R)).
```

Листинг 3: Доступность предмета игроку в несколько шагов

```
1 :- module(evil, [evil_biome/1]).
2
3 evil_biome('Corruption').
4 evil_biome('Crimson').
```

Листинг 4: Злые биомы Террарии

```
1 :- module(ore, [ore/1, ore_power/2, existing/1]).
2
3 ore('Copper').
4 ore('Tin').
5 ore('Iron').
6 ore('Lead').
7 ore('Silver').
8 ore('Tungsten').
9 ore('Gold').
10 ore('Platinum').
11 ore('Meteorite').
12 ore('Demonite').
13 ore('Crimtane').
14 ore('Obsidian').
15 ore('Hellstone').
16 ore('Cobalt').
17 ore('Palladium').
18 ore('Mythril').
19 ore('Orichalcum').
20 ore('Adamantite').
21 ore('Titanium').
22 ore('Chlorophyte').
23 ore('Luminite').
24
25 existing(ore(Ore)) :- ore(Ore).
26
27 ore_power(ore('Copper'), 1).
28 ore_power(ore('Tin'), 1).
29 ore_power(ore('Iron'), 2).
30 ore_power(ore('Lead'), 2).
31 ore_power(ore('Silver'), 3).
32 ore_power(ore('Tungsten'), 3).
33 ore_power(ore('Gold'), 4).
34 ore_power(ore('Platinum'), 4).
35 ore_power(ore('Meteorite'), 4).
36 ore_power(ore('Demonite'), 5).
37 ore_power(ore('Crimtane'), 5).
38 ore_power(ore('Obsidian'), 5).
39 ore_power(ore('Hellstone'), 6).
40 ore_power(ore('Cobalt'), 7).
41 ore_power(ore('Palladium'), 7).
42 ore_power(ore('Mythril'), 8).
43 ore_power(ore('Orichalcum'), 9).
44 ore_power(ore('Adamantite'), 10).
45 ore_power(ore('Titanium'), 11).
```

Листинг 5: Руды Террарии

```
1 :- module(pickaxe, [
2     pickaxe/1,
3     pickaxe_power/2,
4     pickaxe_with_power/1
5 ]).
6
7 pickaxe('Cactus').
8 pickaxe('Copper').
9 pickaxe('Tin').
10 pickaxe('Iron').
11 pickaxe('Lead').
12 pickaxe('Silver').
13 pickaxe('Tungsten').
14 pickaxe('Gold').
```

```

15 pickaxe('Platinum').
16 pickaxe('Nightmare').
17 pickaxe('Deathbringer').
18 pickaxe('Molten').
19 pickaxe('Cobalt').
20 pickaxe('Palladium').
21 pickaxe('Mythril').
22 pickaxe('Orichalcum').
23 pickaxe('Adamantite').
24 pickaxe('Titanium').
25
26 pickaxe_power(pickaxe('Cactus'), 2).
27 pickaxe_power(pickaxe('Copper'), 2).
28 pickaxe_power(pickaxe('Tin'), 2).
29 pickaxe_power(pickaxe('Iron'), 3).
30 pickaxe_power(pickaxe('Lead'), 3).
31 pickaxe_power(pickaxe('Silver'), 4).
32 pickaxe_power(pickaxe('Tungsten'), 4).
33 pickaxe_power(pickaxe('Gold'), 5).
34 pickaxe_power(pickaxe('Platinum'), 5).
35 pickaxe_power(pickaxe('Nightmare'), 6).
36 pickaxe_power(pickaxe('Deathbringer'), 6).
37 pickaxe_power(pickaxe('Molten'), 7).
38 pickaxe_power(pickaxe('Cobalt'), 8).
39 pickaxe_power(pickaxe('Palladium'), 8).
40 pickaxe_power(pickaxe('Mythril'), 9).
41 pickaxe_power(pickaxe('Orichalcum'), 9).
42 pickaxe_power(pickaxe('Adamantite'), 10).
43 pickaxe_power(pickaxe('Titanium'), 10).
44
45 pickaxe_with_power(1).
46 pickaxe_with_power(2).
47 pickaxe_with_power(3).
48 pickaxe_with_power(4).
49 pickaxe_with_power(5).
50 pickaxe_with_power(6).
51 pickaxe_with_power(7).
52 pickaxe_with_power(8).

```

Листинг 6: Кирки Террарии

```

1 :- module(progress, [found/1]).
2
3 :- use_module(pickaxe).
4 :- use_module(ore).
5
6 :- dynamic(found/1).
7
8 % found(evil_biome('Crimson')).
9 %
10 % found(existing(ore('Crimtane'))):-
11 %     found(evil_biome('Crimson')).
12 % found(existing(ore('Demonite'))):-
13 %     found(evil_biome('Corruption')).
14 %
15 % found(existing(ore('Demonite'))).
16 %
17 % found(existing(ore('Tin'))).
18 % found(existing(ore('Iron'))).
19 % found(existing(ore('Silver'))).
20 % found(existing(ore('Gold'))).
21 %
22 % found(existing(ore('Hellstone'))).
23 %
24 % found(ore('Copper')).
25 % found(pickaxe('Copper')).
26 % found(pickaxe('Gold')).
27 % found(pickaxe('Deathbringer')).
28 % found(pickaxe('Molten')).
29 %
30
31 found(pickaxe_with_power(Power)) :-
32     number(Power),

```

```

33     found(pickaxe(Pickaxe)),
34     pickaxe_power(pickaxe(Pickaxe), PickaxePower),
35     (PickaxePower < Power -> false ; true)).

```

Листинг 7: Прогресс игрока

```

1 :- module(requirement, [requirement/2]).
2
3 :- use_module(pickaxe).
4 :- use_module(ore).
5 :- use_module(evil).
6
7 requirement(pickaxe('Copper'), ore('Copper')).
8 requirement(pickaxe('Tin'), ore('Tin')).
9 requirement(pickaxe('Iron'), ore('Iron')).
10 requirement(pickaxe('Lead'), ore('Lead')).
11 requirement(pickaxe('Silver'), ore('Silver')).
12 requirement(pickaxe('Tungsten'), ore('Tungsten')).
13 requirement(pickaxe('Gold'), ore('Gold')).
14 requirement(pickaxe('Platinum'), ore('Platinum')).
15 requirement(pickaxe('Nightmare'), ore('Demonite')).
16 requirement(pickaxe('Deathbringer'), ore('Crimtane')).
17 requirement(pickaxe('Molten'), ore('Hellstone')).
18 requirement(pickaxe('Cobalt'), ore('Cobalt')).
19 requirement(pickaxe('Palladium'), ore('Palladium')).
20 requirement(pickaxe('Mythril'), ore('Mythril')).
21 requirement(pickaxe('Orichalcum'), ore('Orichalcum')).
22 requirement(pickaxe('Adamantite'), ore('Adamantite')).
23 requirement(pickaxe('Titanium'), ore('Titanium')).
24
25 requirement(ore(Ore), pickaxe_with_power(PickaxePower)) :-
26     ore_power(ore(Ore), PickaxePower).
27
28 requirement(ore(Ore), existing(ore(Ore))).

```

Листинг 8: Требования для добычи предметов

4.2 Онтология в Protege

<https://github.com/vityaman-edu/ai-prolog/blob/trunk/res/ai-lab-2.rdf>

4.3 Тестирование и отладка системы, обеспечение ее функциональности и эффективности

Была написана система принятия решений в виде CLI чат-бота на языке Python.

Запросы к базе знаний выполнялись с помощью библиотеки `pyswip`.

<https://github.com/vityaman-edu/ai-prolog/blob/trunk/src/prolog.py>

Также в Python была выражена часть предметной области для функции для построения запросов к серверу Prolog и удобного тестирования.

<https://github.com/vityaman-edu/ai-prolog/blob/trunk/src/dsl.py>

Для разбора ответов сервера Prolog был сделан простенький парсер выражений на основе Python Lark.

<https://github.com/vityaman-edu/ai-prolog/blob/trunk/src/parse.py>

На базе описанного выше удалось спрятать Prolog за слоем бизнес логики.

<https://github.com/vityaman-edu/ai-prolog/blob/trunk/src/terraria.py>

Слой бизнес логики используется слоем принятия и обработки текстовых запросов.

<https://github.com/vityaman-edu/ai-prolog/blob/trunk/src/feedback.py>

Ну и вишенкой на торте стало CLI приложение.

<https://github.com/vityaman-edu/ai-prolog/blob/trunk/src/args.py>

<https://github.com/vityaman-edu/ai-prolog/blob/trunk/src/main.py>

Программное обеспечение всегда страшно разрабатывать без тестов, поэтому без них не обошлось.

Было проведено модульное тестирование парсера.

https://github.com/vityaman-edu/ai-prolog/blob/trunk/src/test_parse.py

Также слегка проверен был слой бизнес-логики.

https://github.com/vityaman-edu/ai-prolog/blob/trunk/src/test_terraria.py

Чтобы быть более уверенным в корректной обработки даже неправильных запросов пользователя была реализована некоторая пародия на фаззинг-тестирование.

https://github.com/vityaman-edu/ai-prolog/blob/trunk/src/test_fuzzing.py

4.4 Пример вывода программы

<https://github.com/vityaman-edu/ai-prolog/blob/trunk/res/example.txt>

5 Оценка и интерпретация результатов

Наша система соответствует заявленным требованиям.

Цель была достигнута - мы разработали CLI чат-бота предоставляющего доступ к базе знаний на языке Prolog, он помогает человеку играть в игру Terraria, советуя, какие шаги предпринять и давая комментарии по поводу прогресса в игре.

В дальнейшем можно расширять базу знаний, добавлять новые аспекты игры. Расширить вариативность как ввода, так и вывода. Сделать бота более гибким и похожим на человека. Также следует исправить существующие проблемы приложения (баги) и повысить тестовое покрытие, доработать фаззинг.

6 Заключение

Разработанное приложение поможет игрокам в Terraria понять правила игры и получить подсказки.

Список литературы

- [1] [https://ru.wikipedia.org/wiki/%D0%9E%D0%BD%D1%82%D0%BE%D0%BB%D0%BE%D0%B3%D0%B8%D1%8F_\(%D0%B8%D0%BD%D1%84%D0%BE%D1%80%D0%BC%D0%B0%D1%82%D0%B8%D0%BA%D0%B0\)](https://ru.wikipedia.org/wiki/%D0%9E%D0%BD%D1%82%D0%BE%D0%BB%D0%BE%D0%B3%D0%B8%D1%8F_(%D0%B8%D0%BD%D1%84%D0%BE%D1%80%D0%BC%D0%B0%D1%82%D0%B8%D0%BA%D0%B0))
- [2] https://ru.wikipedia.org/wiki/%D0%91%D0%B0%D0%B7%D0%B0_%D0%B7%D0%BD%D0%B0%D0%BD%D0%B8%D0%B9
- [3] <https://www.swi-prolog.org/>
- [4] <https://protege.stanford.edu/>
- [5] <https://github.com/yuce/pyswip>