Федеральное государственное автономное образовательное учреждение высшего образования «Национальный исследовательский университет ИТМО»

Факультет программной инженерии и компьютерной техники

Лабораторные работы №1-3 по дисциплине

«**Системы искусственного интеллекта**»

**Выполнил:**

Миху Вадим Дмитриевич

Факультет “ПИиКТ”

Группа: P33301

**Преподаватель:**

Кугаевских Александр Владимирович

Вариант Diablo II LOD



Санкт-Петербург, 2023 г.

## Оглавление

[Задание 2](#_Toc2859)

[Анализ требований 2](#_Toc2860)

[Изучение основных концепций и инструментов 2](#_Toc2861)

[Реализация системы искусственного интеллекта на Prolog 4](#_Toc2862)

[Оценка и интерпретация результатов 4](#_Toc2863)

[Заключение 5](#_Toc2864)

# Задание

Требуется создать базу знаний в языке программирования Prolog и реализовать набор запросов, используя эту базу знаний. Задача направлена на развитие навыков работы с фактами, предикатами, и правилами в логическом программировании.

Целью второй лабораторной работы является знакомство со средой разработки онтологий Protege и перевод базы знаний, созданной в предыдущей лабораторной работе в онтологическую форму в Protege.

Целью третьей лабораторной работы является разработка программы, которая будет использовать базу знаний или онтологию для предоставления рекомендаций на основе введенных пользователем данных. (Knowledge-based support system)

# Анализ требований

Система поддержки принятия решений должна:

· Обрабатывать ввод пользователя.

· Иметь возможность к расширению.

· Инициализировать базу знаний самостоятельно.

Требования к базе знаний:

· База знаний должна основываться на информации про игру Diablo II.

· Должна состоять как из фактов, так и из правил.

· Должна уточнять требования к онтологии, включая определение классов, свойств и отношений между сущностями.

# Изучение основных концепций и инструментов

### 1. Базы знаний и онтологии

* Базы знаний представляют собой совокупность фактов, правил и информации, которая может быть использована для автоматической обработки данных и принятия решений. Они широко используются в экспертных системах, системах поддержки принятия решений и других областях искусственного интеллекта.
* Онтологии – это формализованные описания понятий и их взаимосвязей в определенной области знаний. Они позволяют компьютерам понимать и интерпретировать данные, а также делать выводы на основе этого понимания. Онтологии играют важную роль в семантическом вебе и интеллектуальных агентах.

### 2. Prolog и его возможности

Prolog (Programming in Logic) – это декларативный язык программирования, основанный на логике предикатов. Он широко используется для решения задач, связанных с логическим выводом и обработкой баз знаний. Преимущества Prolog включают:

· Декларативность: Prolog позволяет описывать задачи и отношения между данными декларативно, что делает код более легким для понимания и сопровождения.

· Логический вывод: Язык основан на формальной логике и предоставляет мощные средства для логического вывода. Это позволяет создавать системы, способные делать сложные логические рассуждения.

· Обработка баз знаний: Prolog хорошо подходит для работы с базами знаний и онтологиями, так как его синтаксис и семантика позволяют легко представлять и манипулировать знаниями.

3. Инструменты и библиотеки для работы с базами знаний и онтологиями на Prolog

Для разработки систем искусственного интеллекта на Prolog существует множество инструментов и библиотек, которые облегчают работу с базами знаний и онтологиями:

SWI-Prolog – это одна из наиболее популярных реализаций языка, которая предоставляет множество встроенных средств для работы с базами знаний, включая графический интерфейс для создания и редактирования онтологий.

Существуют библиотеки, расширяющие возможности Prolog для выполнения логического вывода, что делает его более мощным инструментом для разработки интеллектуальных систем, в том числе библиотека swiplserver для языка python.

# Реализация системы искусственного интеллекта на Prolog

% Факты с одним аргументом

hero(dragon\_knight).

hero(earthshaker).

hero(invoker).

hero(juggernaut).

hero(mirana).

hero(sven).

hero(tidehunter).

hero(windranger).

hero(zeus).

hero(antimage).

hero(bristleback).

hero(drow\_ranger).

hero(pudge).

hero(phantom\_assassin).

hero(legion\_commander).

hero(lion).

hero(crystal\_maiden).

hero(lich).

hero(ogre\_magi).

hero(luna).

item(clarity).

item(tango).

item(basher).

item(eul).

item(rapier).

item(tarasque).

% Факты с двумя аргументами

sister(dragon\_knight,mirana).

sister(invoker,luna).

brother(crystal\_maiden,lion).

brother(windranger,tidehunter).

ability(dragon\_knight,breath\_fire).

ability(dragon\_knight,dragon\_tail).

ability(dragon\_knight,dragon\_blood).

ability(dragon\_knight,fireball).

ability(invoker,quas).

ability(invoker,wex).

ability(invoker,exort).

ability(invoker,invoke).

ability(juggernaut,blade\_fury).

ability(juggernaut,omnislash).

property(lich,intelligent).

property(invoker,intelligent).

property(luna,agility).

property(pudge,strength).

cost(tango,100).

cost(clarity,150).

cost(basher,1000).

cost(eul,2000).

cost(rapier,6000).

cost(tarasque,6500).

% Правила

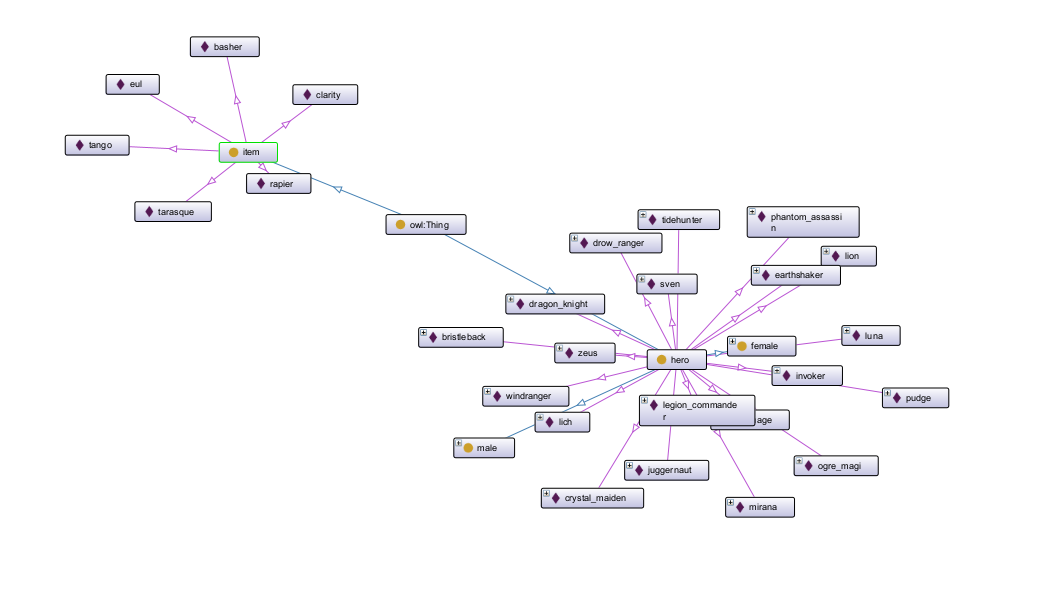
can\_use\_ability(A, B) :- hero(A), ability(A, B).

affordable\_item(Item, Gold) :- item(Item), cost(Item, Cost), Gold >= Cost.

normal\_item(Item) :- item(Item), cost(Item, Cost), Cost < 2000.

cheap\_item(Item) :- item(Item), cost(Item, Cost), Cost < 200.

**База знаний Protégé**

****

**Система принятия решения на Python**

**from pyswip import Prolog**

**prolog = Prolog()**

**prolog.consult('main.pl')**

**def recommend\_heroes(preferred\_property):**

**return list(prolog.query(f"property(X, {preferred\_property})"))**

**def recommend\_items(gold\_available):**

**return list(prolog.query(f"affordable\_item(X, {gold\_available})"))**

**def parse\_input(user\_input):**

**user\_preferences = user\_input.split(', ')**

**preferred\_property = user\_preferences[1].split(': ')[1]**

**gold\_available = int(user\_preferences[2].split(': ')[1])**

**return preferred\_property, gold\_available**

**def give\_recommendations(user\_input):**

**preferred\_property, gold\_available = parse\_input(user\_input)**

**recommended\_heroes = recommend\_heroes(preferred\_property)**

**recommended\_items = recommend\_items(gold\_available)**

**return recommended\_heroes, recommended\_items**

**# Пример использования**

**user\_input = 'Мне 13 лет, предпочитаемое свойство: intelligent, доступное золото: 100'**

**recommendations = give\_recommendations(user\_input)**

**print(recommendations)**

# Оценка и интерпретация результатов

Система соответствует всем поставленным требованиям.

В дальнейшем можно дополнять базу знаний, прописывать новые связи. Также добавить больше шаблонов для пользовательского ввода и сделать ввод более гибким.

# Заключение

Базы знаний и онтологии могут предоставить удобный функционал для реализации некоторых задач, которые при других подходах имели бы гораздо более запутанную реализацию.