Факультет программной инженерии и компьютерной техники

Лабораторная работа №1

«Создание базы знаний и выполнение запросов в Prolog»

по дисциплине «Системы искусственного интеллекта»

Выполнили:

Студент группы P3332

Чмурова М.В.

Преподаватель:

Бессмертный Игорь Александрович

Санкт-Петербург

2024

Оглавление

Введение

Цель проекта — создать базу знаний в Prolog. Смоделировать взаимодействие объектов внутри игровой вселенной, и использовать базу знаний для выполнения запросов, что позволит эффективно обрабатывать знания об игровых персонажах, их предметах и местоположении.

Данный проект предоставляет базовый, но мощный пример использования логического программирования для построения искусственного интеллекта и систем поддержки принятия решений, особенно в сфере видеоигр.

Анализ требований

Необходимо создание и поддержка базы знаний, описывающей игровые объекты, их характеристики и отношения между ними. Нужно ввести логические правил, моделирующих события, действия и возможности персонажей, реализовать запросы к базе знаний для оценки различных игровых ситуаций и добавить возможность изменения состояния объектов, что моделирует динамическое поведение мира.

База знаний должна включать в себя не менее 20 фактов с одним аргументом, 10-15 фактов с двумя аргументам, которые дополняют и показывают связь с другими фактами и 5-7 правил. Факты могут описывать объекты, их свойства и отношения между ними. Факты 2 и более аргументами могут описывать различные атрибуты объектов, а правила - логические законы и выводы, которые можно сделать на основе фактов и предикатов.

Обзор основных концепций баз знаний и онтологий

Концепции баз знаний:

1. Факт представляет собой утверждение о свойствах объекта или отношении между объектами. Например, survivor(dwight) означает, что "Дуайт — Выживший".
2. Предикаты являются основой выражений в Prolog и описывают отношения. Например, предикат location(dwight, house) связывает объект с его местоположением.
3. Правила используются для выведения новой информации на основе имеющихся фактов и отношений. Например, правило can_repair(dwight, generator1) проверяет, может ли персонаж починить генератор, находясь с ним в одном месте.

Реализация системы искусственного интеллекта

Реализованные факты:

```
% Выжившие (survivors)
survivor(dwight).
survivor(meg).
survivor(claudette).
survivor(jake).
survivor(nea).

% Убийцы (killers)
killer(trapper).
killer(wraith).
killer(hillbilly).
killer(nurse).
killer(michael_myers).

% Существующие места на карте
place(house).
place(barn).
place(forest).
place(street).

% Предметы Выживших (items)
item(flashlight).
item(medkit).
item(toolbox).
item(map).
item(key).

% Генераторы на карте (generators)
generator(generator1).
generator(generator2).
generator(generator3).
```

Реализованные предикаты:

```
% Связь пердметов, которые используют Выжившие:
has_item(dwight, medkit).
has_item(meg, flashlight).
has_item(claudette, toolbox).
has_item(jake, map).
has_item(nea, key).

% Состояние генераторов:
generator_state(generator1, broken).
generator_state(generator2, broken).
generator_state(generator3, broken).

% Местоположение Убийц, Выживших и генераторов на карте:
location(dwight, house).
location(meg, barn).
location(claudette, forest).
location(jake, barn).
location(nea, street).

location(trapper, barn).
location(wraith, forest).
location(michael_myers, house).

location(generator1, house).
location(generator2, barn).
location(generator3, forest).
```

Реализованные правила:

```
% 1 Правила на проверку, что Выживший может починить генератор
can_repair(Survivor, Generator) :-
    survivor(Survivor),
    location(Survivor, Location),
    location(Generator, Location),
    generator_state(Generator, broken).
```

```
% 2 Проверка, что Выживший может сбежать
can_escape(Survivor) :-
    survivor(Survivor),
    generator_state(generator1, repaired),
    generator_state(generator2, repaired),
    generator_state(generator3, repaired).

% 3 Проверка, что Убийца может обнаружить Выжившего
can_detect(Killer, Survivor) :-
    killer(Killer),
    survivor(Survivor),
    location(Killer, Location),
    location(Survivor, Location).

% 4 Изменение состояние генератора после починки
repair_generator(Generator) :-
    retract(generator_state(Generator, broken)),
    assert(generator_state(Generator, repaired)).

% 5 Изменения состояния генератора после поломки
damage_generator(Generator) :-
    retract(generator_state(Generator, repaired)),
    assert(generator_state(Generator, broken)).

% 6 Проверка, что Выживший может использовать предмет
can_use_item(Survivor, Item) :-
    survivor(Survivor),
    has_item(Survivor, Item).
```

Составленная база знаний отвечает всем описанным требованиям, содержит в себе правила, предикаты и правила.

Примеры запросов

```
location(X, house), survivor(X).
% Определение Выживших, который находятся в доме

survivor(X), \+ has_item(X, flashlight).
% Определение Выживших, у которых нет фонарика

generator_state(Generator, broken).
% Определение всех сломанных генераторов

can_use_item(X, medkit).
% Определение персонажей, который могут использовать предмет
medkit

can_repair(X, generator1).
% Определение кто из выживших может починить генератор1

can_detect(trapper, claudette).
%Определение может ли Убийца обнаружить Выжившиего

repair_generator(generator1).
% Изменения состояния генератора на починенный

damage_generator(generator2).
% Изменения состояние генератора на сломанный

can_escape(dwight).
% Проверка может ли сбежать выживший
```

Пример выполнения некоторых запросов к базе знаний:

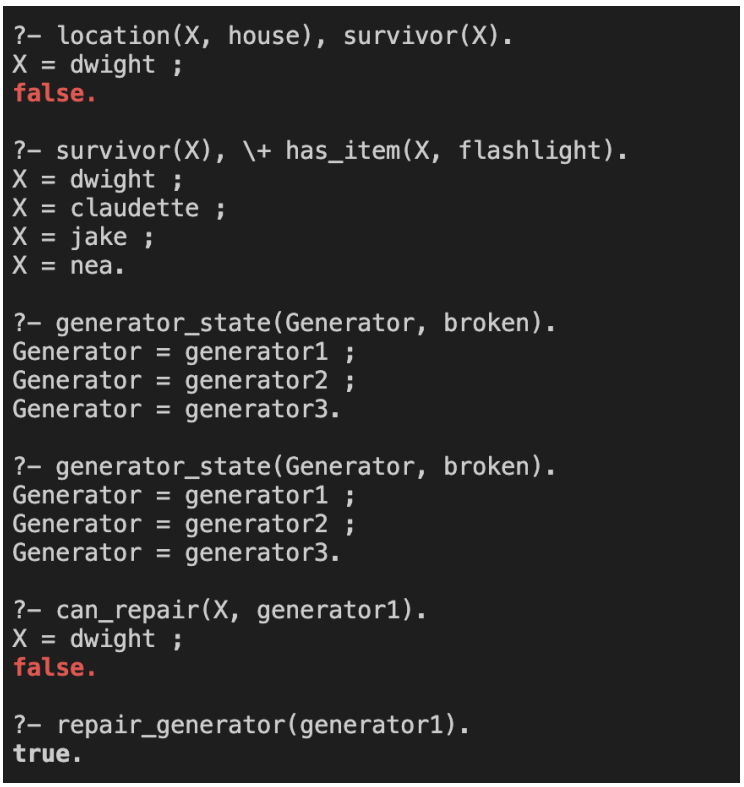

Рисунок 1. Выполнение запросов к базе знаний

База знаний была протестирована с использованием различных запросов, включающих логические операторы, переменные и правила для оценки корректности и гибкости системы.

Были использованы простые и сложные запросы, что позволило убедиться в правильности работы всех условий, особенно проверок с изменением состояний генераторов.

Заключение

Преимущества этой системы заключаются в её гибкости и возможностях логического вывода, которые позволяют моделировать сложные взаимодействия и автоматизировать принятие решений. Система может применяться в различных игровых жанрах, требующих сложного поведения NPC. База знаний также позволяет легко расширять логику и добавлять новые объекты, правила и запросы, что делает её полезной для создания интерактивных сюжетов и принятия решений в виртуальных мирах.