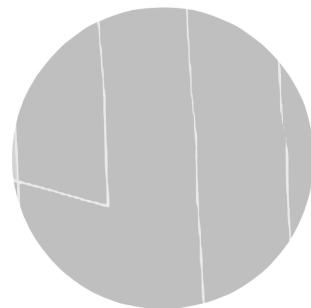




МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РФ

Федеральное государственное автономное  
образовательное учреждение высшего образования  
“Национальный исследовательский университет ИТМО”

**ФАКУЛЬТЕТ ПРОГРАММНОЙ ИНЖЕНЕРИИ  
И КОМПЬЮТЕРНОЙ ТЕХНИКИ**



## МОДУЛЬ №1

по дисциплине  
“Системы искусственного интеллекта”

*Работу выполнил:*  
Студент группы Р3311  
**Болорболд Аригуун**  
*Преподаватель:*

**Кугаевских Александр Владимирович**

г. Санкт-Петербург  
2024 г.

# 1 Введение

## Описание целей проекта и его значимости

Органическая химия — раздел химии, изучающий структуру, свойства и методы синтеза соединений углерода с другими химическими элементами, относящихся к органическим соединениям.

Значимость первой лабораторной работы определяется созданием базы знаний и онтологии органических соединений с помощью Prolog и Protege. База знаний может использоваться для разработки системы поддержки принятия решения.

Значимость второй лабораторной работы определяется разработкой программы, которая будет использовать базу знаний или онтологию для предоставления рекомендаций на основе введенных пользователем данных. (Knowledge-based support system).

## 2 Анализ требований

### Определение основных требований к системе поддержки принятия решений

1. Точность (получение подробной информации о конкретных молекулах).
2. Персонализация (индивидуальный подход к каждому пользователю).
3. Эффективность и быстрое действие.
4. Простота (простота в использовании, интуитивно понятный интерфейс).
5. Вариативность (для разных входных данных, разный результат).
6. Реализация разных функций получения следующей информации:

- Классификация;
- Молярная масса;
- Плотность;
- Температуры переходов фаз (кипения, плавления, сублимации);
- Растворимость в воде;
- Баллы по стандарту [NFPA 704](#).

### Выявление требований к базе знаний и онтологии для представления знаний

Требования к базе знаний и онтологиям определяются необходимостью хранения корректной, структурированной информации для заданной предметной области. Необходимо продумать четкую иерархию сущностей/фактов. Говоря о базе знаний, выделить четкие факты, предикаты, правила. Работая с онтологией, необходимо четко определить сущности, свойства.

## 3 Изучение основных концепций и инструментов

- Обзор основных концепций баз знаний и онтологий.

Базы знаний и онтологии — это два понятия в области искусственного интеллекта и компьютерных наук. Обе концепции используются для описания знаний и информации, но имеют разные подходы к этому.

- Базы знаний — это системы, которые хранят информацию и знания, которые могут быть использованы для решения задач. Они обычно содержат факты, правила, процедуры и другие элементы, необходимые для решения задач. Базы знаний используются в различных областях, таких как медицина, финансы, право и т.д.
- Онтологии — это формальные модели, которые описывают концептуальные категории и отношения между ними. Они используются для описания знаний в языке, который может быть понят людьми и машинами. Онтологии используются для создания более точных и эффективных систем поиска, классификации и анализа данных.

Основными принципами баз знаний являются:

1. Декомпозиция знаний на факты, правила и процедуры.
2. Построение моделей знаний на основе логических и математических принципов.
3. Использование методов интеллектуального анализа данных для решения задач.

Основными принципами онтологий являются:

1. Описание концептуальных категорий и отношений между ними.
2. Использование формальных языков для описания знаний.
3. Создание стандартных моделей для обмена и интеграции данных.

Обе концепции имеют свои преимущества и недостатки, и их выбор зависит от конкретных задач и потребностей. Базы знаний используются для решения конкретных задач, а онтологии — для описания общих концептуальных категорий и создания стандартных моделей для обмена данными.

- Изучение Prolog и его возможностей для разработки систем искусственного интеллекта.

Prolog — это язык программирования общего назначения. Он обладает возможностями для создания эффективных алгоритмов и моделей машинного обучения. Некоторые из наиболее полезных функций и возможностей Prolog:

1. Логический вывод. Prolog поддерживает логические операции, такие как AND, OR, NOT, EQUALS и LT, GT, LE, BETWEEN и IN, которые могут использоваться для выполнения различных типов логических операций. Это позволяет создавать эффективные модели логического вывода и обработки естественного языка.
2. Моделирование. Prolog предоставляет мощные инструменты для создания и манипулирования моделями машинного обучения. Он поддерживает различные типы моделей, включая линейную регрессию, деревья решений, нейронные сети и многое другое. Prolog также поддерживает возможность создания собственных моделей с помощью пакетов, таких как PL/SQL и Symbolic Expression Language (SEL).

3. Обработка естественного языка. Prolog поддерживает обработку естественного языка, такую как токенизация, лемматизация, стемминг и синтаксический анализ текста. Это позволяет создавать эффективные модели обработки естественного языка и анализировать текстовые данные.
  4. Алгоритмическое программирование. Prolog поддерживает различные типы алгоритмического программирования, такие как задачи коммивояжера, сортировка слиянием и быстрая сортировка. Это позволяет создавать эффективные алгоритмы для решения задач, которые требуют сложных структур данных и вычислительных ресурсов.
  5. Интеграция с другими языками программирования. Prolog поддерживает интеграцию с различными языками программирования, такими как Python, Java и C++. Это позволяет разработчикам интегрировать свои модели машинного обучения и алгоритмы в другие проекты и среды разработки. Prolog предлагает мощный и гибкий инструмент для разработки систем искусственного интеллекта. Он обеспечивает высокую производительность, эффективность и удобство использования, а также поддерживает широкий спектр функциональности, которая может быть использована для создания эффективных моделей машинного обучения и обработки естественного языка.
- Ознакомление с инструментами и библиотеками, подходящими для работы с базами знаний и онтологиями на Prolog. Существует множество инструментов и библиотек, которые могут помочь вам работать с базами знаний и онтологиями на языке Prolog. Вот несколько примеров:
    1. PL/SQL: PL/SQL — это язык программирования, который широко используется для управления реляционными базами данных. Он предоставляет удобные средства для создания, изменения и удаления таблиц, а также для выполнения запросов к базе данных. Можно использовать PL/SQL для создания моделей баз данных, таких как деревья решений, нейронные сети и графы, а также для анализа и обработки данных.
    2. SQLAlchemy: SQLAlchemy — это фреймворк для создания и управления базами данных на языке SQL. Он предоставляет удобный интерфейс для взаимодействия с реляционными базами данных, а также для создания и управления базами данных на языке SQL. SQLAlchemy также поддерживает работу с онтологиями, позволяя создавать и управлять базами знаний на языке SQL.
    3. TensorFlow: TensorFlow — это библиотека машинного обучения, которая предоставляет мощные инструменты для создания и обучения нейронных сетей. Она поддерживает работу с базами знаний, позволяя создавать и управлять базами знаний на языке SQL. TensorFlow также поддерживает работу с онтологиями, позволяя создавать и управлять базами знаний на языке SQL.
    4. OWL: OWL — это язык разметки, который используется для описания свойств и отношений между сущностями в базе знаний. Он предоставляет удобный интерфейс для создания и управления базами знаний на языке SQL. OWL также поддерживает работу с онтологиями, позволяя создавать и управлять базами знаний на языке SQL.

5. MongoDB: MongoDB — это популярная NoSQL база данных, которая предоставляет мощные инструменты для работы с данными. Она поддерживает работу с базами знаний, позволяя создавать и управлять базами знаний на языке SQL. MongoDB также поддерживает работу с онтологиями, позволяя создавать и управлять базами знаний на языке SQL.

## 4 Реализация системы искусственного интеллекта на Prolog

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

Факты базы знаний:

Факт	Описание
<i>organic_compound</i> (X)	X — органическое вещество
<i>chemical_class</i> (X, Y)	X — органическое вещество, Y — классификация
<i>molar_mass</i> (X, Y)	X — органическое вещество, Y — молярная масса
<i>density</i> (X, Y)	X — органическое вещество, Y — плотность
<i>temperatures</i> (X, Y, Z)	X — органическое вещество, Y — температура плавления, Z — температура кипения
<i>vapor_pressure</i> (X, Y)	X — органическое вещество, Y — давление насыщенного пара
<i>water_solubility</i> (X, Y)	X — органическое вещество, Y — растворимость в воде
<i>nfpa</i> (X, H, F, R, S)	X — органическое вещество, H — опасность здоровью, F — огнеопасность, R — реактивность, S — специальное предупреждение

Правила (Prolog):

- % Правило 1. Список всех соединений:

```
list_of_compounds(Compounds) :-
    findall(Compound, organic_compound(Compound), Compounds).
```

- % Правило 2. Проверка на изомерность молекул:

```
is_isomeric(X, Y) :-
    molar_mass(X, M_mol1),
    molar_mass(Y, M_mol2),
    X \= Y,
    M_mol1 = M_mol2.
is_isomeric(Y, X) :-
    molar_mass(Y, M_mol1),
    molar_mass(X, M_mol2),
    Y \= X,
    M_mol1 = M_mol2.
```

- % Правило 3. Проверка на лёгкость молекул:

```
is_lightweight(X) :-
    molar_mass(X, M_mol), M_mol=<90.
```

- % Правило 4. Проверка на плотность веществ по сравнению с водой (1000кг/м<sup>3</sup>):

```
is_dense(X) :-
    density(X, Rho), Rho > 1000.
```

- % Правило 5. Проверка на агрегатное состояние вещества при комнатной температуре (25°C):

```
state_of_matter(X) :-
    temperatures(X, T_1, T_2),
    ( T_1 < 25, T_2 > 25
    -> State = 'жидким'
    ; T_1 > 25
    -> State = 'твёрдым'
    ; T_2 < 25
    -> State = 'газообразным'
    ),
    format('Вещество является ~w.', [State]).
```

- % Правило 6. Проверка на летучесть вещества (сравнивается с водой при 20°C):

```
is_volatile(X) :-
    vapor_pressure(X, P),
    ( number(P)
    -> P > 2.3388
    ; P = 'Empty'
    ).
```

- % Правило 7. Проверка на растворимости в воде:

```
% - <0,1: нерастворимый;
% - 0,1-1: малорастворимый;
% - >1: растворимый;
% - 'miscible': неограниченно.
```

```
solubility(X) :-
    water_solubility(X, G),
    ( number(G)
    -> ( G<0.1
    -> Sol = 'нерастворимым'
    ; G>=0.1; G<1
    -> Sol = 'малорастворимым'
    ; G>=1
    -> Sol = 'растворимым'
    )
    )
```



```
; Sol = 'неограниченно растворимым'
),
format('Вещество является ~w в воде.', [Sol]).
```

```
– % Правило 8. Проверка безопасности вещества по токсичности:
% 0 — безопасный;
% 1 — вызывает раздражение;
% 2 — вред при интенсивном и/или длительном воздействии;
% 3 — вред при кратковременном воздействии;
% 4 — смерть при очень кратковременном воздействии.

% Проверка безопасности вещества по огнеопасности:
% 0 — негорючее;
% 1 — трудногорючее;
% 2 — отн. высокая температура воздуха;
% 3 — при температуре воздуха;
% 4 — ниже температуры воздуха.

% Проверка безопасности вещества по стабильности:
% 0 — стабильный;
% 1 — обычно стабильный, но может стать неустойчивым при повышенных температуре и давлении;
% 2 — неустойчивый при повышенных температуре и давлении;
% 3 — может детонировать при специфичных условиях;
% 4 — может детонировать.

nfpa_hazard(X, H, F, R) :-
    nfpa(X, H, F, R, _).

– % Правило 9. Проверка безопасности вещества по специальной критерии:
% SA — удушающий газ;
% COR — коррозионное вещество;
% ACID — кислота;

list_of_compounds(Compounds) :-
    findall(Compound, organic_compound(Compound), Compounds).

– % Правило 1. Список всех соединений:

nfpa_special_hazard(X) :-
    nfpa(X, _, _, _, S),
    ( S = 'SA'
    -> Factor = 'удушающим газом'
    ; S = 'COR'
```

```

-> Factor = 'коррозивным'
; S = 'ACID'
-> Factor = 'кислотой'
; S = 'Empty'
-> writeln('Вещество не имеет особенной характеристики.'),
fail
),
format('Вещество является ~w (~w).', [Factor, S]).

```

– % Правило 10. Средний балл безопасности по NFPA:

```

nfpa_avg(X, Avg) :-
    nfpa(X, H, F, R, _),
    Avg is (H+F+R)/3.

```

– % Правило 11. Список веществ со специальными обозначениями:

```

list_of_specials(X_list, S) :-
    findall(X, nfpa(X, _, _, _, S), X_list).

```

– % Правило 12. Сравнение безопасности веществ по их средним значениям по NFPA:

```

nfpa_compare(X, Y) :-
    nfpa_avg(X, Avg_1), nfpa_avg(Y, Avg_2),
    ( Avg_1 > Avg_2
    -> format('~w опаснее, чем ~w.', [X, Y])
    ; Avg_1 < Avg_2
    -> format('~w опаснее, чем ~w.', [Y, X])
    ; format('~w эквивалентен ~w.', [X, Y])
    ).

```

– % Правило 13. Список летучих веществ:

```

list_of_volatiles(Vol) :-
    findall(V, is_volatile(V), Vol).

```

– % Правило 14. Принадлежность соединения к химическому классу:

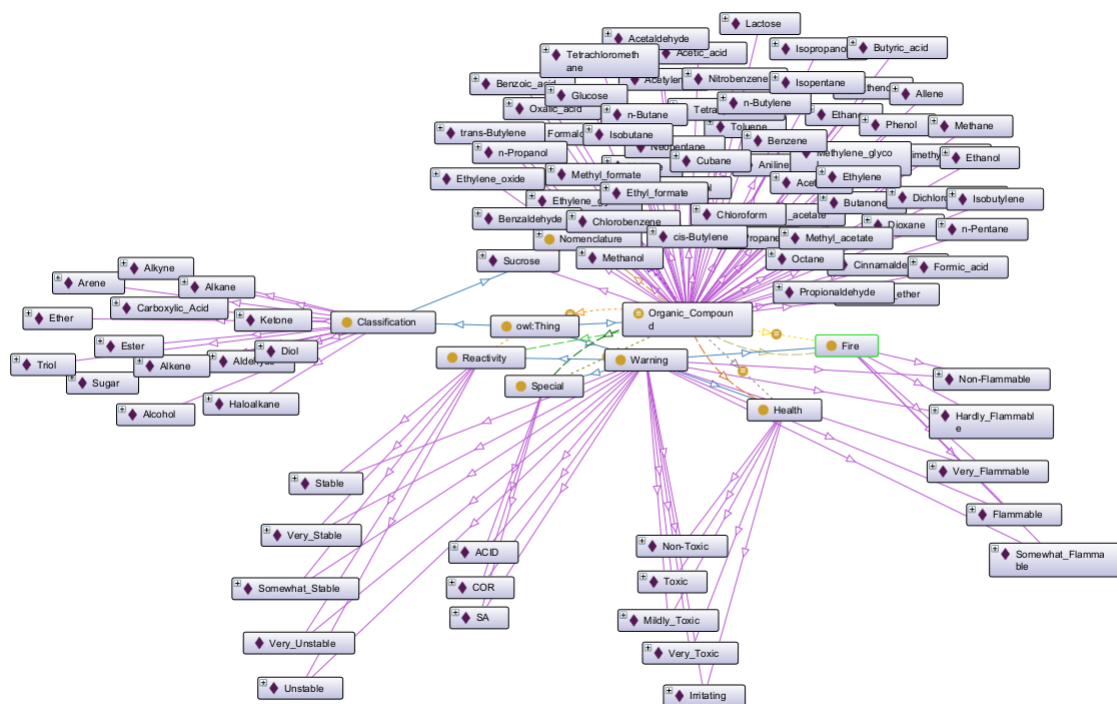
```

belongs_to_class(X, Y) :-
    chemical_class(X, Y).

```

Граф онтологии (Entities)





Запросы в онтологию:

```

Organic_Compound
hasFireWarning value Flammable
hasWaterSolubility value 100000000f
hasMeltingPoint only { 9f, 92f }
hasWaterSolubility only xsd:float[ <= 1261.0f]
Organic_Compound and hasDensity only xsd:float[ >= 10.1]
  
```

- Тестирование и отладка системы, обеспечение её функциональности и эффективности.

Тесты (Запросы к БЗ):

<i>molar_mass('Neopentane', X)</i>	<i>X = 72.151</i>
<i>temperatures('Cubane', X, Y)</i>	<i>X = 133.5, Y = 161.6</i>
<i>(temperatures(X, T<sub>1</sub>, T<sub>2</sub>), T<sub>1</sub> &lt; 25, T<sub>2</sub> &gt; 25; nfpa(X, H, _, _, _), H &gt;= 3)</i>	<i>T<sub>1</sub> = -130.5, T<sub>2</sub> = 36.1, X = ' n - Pentane, ...'</i>
<i>organic_compound(Compound), vapor_pressure(Compound, V), V = 'Empty', nfpa(Compound, H, _, _, _), H =&lt; 1</i>	<i>Compound = ' Allene', H = 0, V = ' Empty', ...</i>
<i>organic_compound(Compound), water_solubility(Compound, 'miscible')</i>	<i>Compound = ' Methane', ' Ethane', ' Propane', ...</i>
<i>molar_mass(X, 72.151)</i>	<i>X = ' n - Pentane', X = ' Isopentane', X = ' Neopentane'</i>
<i>nfpa(X, _, _, R, _), R &gt;= 3</i>	<i>R = 3, X = ' Allene'; R = 3, X = ' Acetylene'; R = 3, X = ' Acetaldehyde'; R = 3, X = ' Ethyleneoxide'</i>
<i>list_of_specials(X, ' SA')</i>	<i>X = ['Methane', ' Ethane', ' n - Butane']</i>
<i>is_volatile(X), nfpa_avg(X, 2)</i>	<i>X = ' Methane', X = ' Propane', X = ' Tetrahydrofuran', X = ' Dioxane'</i>

Тесты (запросы в онтологию):


**Query (class expression)**

hasMeltingPoint only { 9f, 92f }

Execute Add to ontology

**Query results**

Instances (1 of 1)

 **Ethenol**

### Query (class expression)

hasFireWarning **value** Flammable

Execute

Add to ontology

### Query results

Instances (15 of 15)

- ◆ Acetone
- ◆ Butanone
- ◆ Chlorobenzene
- ◆ Dioxane
- ◆ Ethanol
- ◆ Ethyl\_acetate
- ◆ Ethyl\_formate
- ◆ Isopropanol
- ◆ Methanol
- ◆ Methyl\_acetate
- ◆ Octane
- ◆ Propionaldehyde
- ◆ Tetrahydrofuran
- ◆ Toluene
- ◆ n-Propanol

### Query (class expression)

hasWaterSolubility **value** 10000000f

Execute

Add to ontology

### Query results

Instances (16 of 16)

- ◆ Acetaldehyde
- ◆ Acetic\_acid
- ◆ Acetone
- ◆ Butyric\_acid
- ◆ Dioxane
- ◆ Ethanol
- ◆ Ethenol
- ◆ Ethylene\_glycol
- ◆ Ethylene\_oxide
- ◆ Formic\_acid
- ◆ Glycerol
- ◆ Isopropanol
- ◆ Methanol
- ◆ Methylene\_glycol
- ◆ Tetrahydrofuran
- ◆ n-Propanol

**Query (class expression)**

hasWaterSolubility **only** xsd:float[ <= 1261.0f]

**Query results**

Instances (37 of 37)

- ◆ Acetylene
- ◆ Allene
- ◆ Aniline
- ◆ Benzaldehyde
- ◆ Benzene
- ◆ Benzoic\_acid
- ◆ Butanone
- ◆ Chlorobenzene
- ◆ Chloroform
- ◆ Cinnamaldehyde
- ◆ Cubane
- ◆ Dichloromethane
- ◆ Diethyl\_ether
- ◆ Dimethyl\_ether
- ◆ Ethyl\_acetate
- ◆ Ethyl\_formate
- ◆ Ethylene
- ◆ Formaldehyde
- ◆ Glucose
- ◆ Isobutane
- ◆ Isobutylene

**Query (class expression)**

Organic\_Compound **and** hasDensity **only** xsd:float[ >= 10.1]

**Query results**

Instances (49 of 49)

- ◆ Acetaldehyde
- ◆ Acetic\_acid
- ◆ Acetone
- ◆ Aniline
- ◆ Benzaldehyde
- ◆ Benzene
- ◆ Benzoic\_acid
- ◆ Butanone
- ◆ Butyric\_acid
- ◆ Chlorobenzene
- ◆ Chloroform
- ◆ Cinnamaldehyde
- ◆ Cubane
- ◆ Dichloromethane
- ◆ Diethyl\_ether
- ◆ Dioxane
- ◆ Ethanol
- ◆ Ethenol
- ◆ Ethyl\_acetate
- ◆ Ethyl\_formate
- ◆ Ethylene\_glycol

## 5 Оценка и интерпретация результатов

- Примеры запросов для БЗ и онтологии.

Примеры запросов в БЗ и онтологию продемонстрированы выше, в тестировании.

- Оценка соответствия системы поставленным требованиям и достижению целей проекта.

Я доволен реализацией системы, так как (слишком) старался разработать максимально комплексный предсказатель.

- Интерпретация результатов и описание дальнейших возможностей развития и улучшения системы.

Полученные результаты можно интерпретировать как ИИ-помощника, действующего на основе онтологии Protege. Зона роста: интеграция в ин-

формационные системы, посвященные органическим молекулам, постоянное пополнение новыми знаниями.

## 6 Заключение

- Описание преимуществ и потенциальных применений разработанной системы искусственного интеллекта на базе Prolog, баз знаний и онтологий.

На данный момент система готова к использованию. Преимущество - быстрота, надежность, индивидуальный подход к каждому пользователю. Потенциальное применение разработанной системы обусловлено спросом со стороны пользователя. Данную систему можно использовать в любой информационно-формационной системе, посвященной тематике органической химии.

