ОДЕССКИЙ НАЦИОНАЛЬНЫЙ УНИВЕРСИТЕТ ИМ. И. И. МЕЧНИКОВА

ИНСТИТУТ МАТЕМАТИКИ, ЭКОНОМИКИ И МЕХАНИКИ

**Лабораторная работа № 1**

**«Представление знаний семантическими сетями»**

Выполнил

Студент 4 курса ФИТ

Группы ”Сервер”

Поступальский Олег

Одесса 2011

# Цель работы

Разработка модельной программной системы представления знаний на основе семантической сети (СС).

# Задания к лабораторной работе

* Разработать структуры данных для хранения в памяти компьютера списка записей (таблицы) объектов, типов отношений и связей между объектами (не более 100 экземпляров для каждого типа записей).
* Разработать программу, обеспечивающую выполнение следующих функций:
* ввод из текстового файла и сохранение в памяти модели представления знаний СС (см. пример в Приложении 2);
* вывод на экран по запросу списка всех объектов, связей и отношений между объектами;
* ввод с клавиатуры запроса предопределённого формата к базе знаний СС, поиск ответа и вывод его на экран (запрос и ответ должны быть преобразованы в читабельную форму).

# Описание структур данных и основных функций

Так как семантическая сеть имеет много общего с графом, поэтому я использовал для представления семантической сети в своей программе библиотеку классов, разработанную мной, которая содержит необходимые для работы с графами классы. Зависимость классов видна на следующей диаграмме.

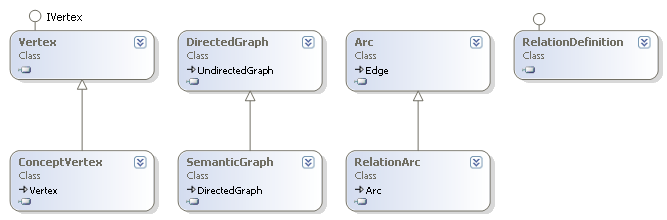


Рис 1. Диаграмма классов SemanticGraph, его компонентов и родительских классов.

Добавив в элементы графа необходимые свойства для представления необходимой семантической сети. Семантический граф также имеет метод поиска логических связей между вершинами, что очень помогает при решении задач семантической сети. Метод описан ниже.

public RelationArc FindRelation(int conceptId1, int conceptId2)

{

var c1 = this.GetConceptById(conceptId1);

var c2 = this.GetConceptById(conceptId2);

var r = this[c1.Value, c2.Value] as RelationArc;

if (r != null)

return r;

foreach (RelationArc arc in this.GetRelations(conceptId1))

{

if (arc.RelationDefinition.IsInheritable())

{

var rArc = this.FindRelation((arc.Head as ConceptVertex).Id, conceptId2);

if (rArc != null)

return rArc;

}

else

{

if (arc.RelationDefinition.IsTransitive())

{

var rArc = this.FindTransitiveRelation((arc.Head as ConceptVertex).Id, conceptId2, arc.Id);

if (rArc != null)

return rArc;

}

}

}

return null;

}

private RelationArc FindTransitiveRelation(int conceptId1, int conceptId2, int relationId)

{

var c1 = this.GetConceptById(conceptId1);

var c2 = this.GetConceptById(conceptId2);

var r = this[c1.Value, c2.Value] as RelationArc;

if (r != null && r.Id == relationId)

return r;

foreach (RelationArc arc in this.GetRelations(conceptId1))

{

if (relationId == arc.Id)

{

var rArc = this.FindTransitiveRelation((arc.Head as ConceptVertex).Id, conceptId2, relationId);

if (rArc != null)

return rArc;

}

}

return null;

}

Для работы с семантическим графом был создан класс SemanticNerwork (Семантическая сеть), в котором есть всего один метод «отвечающий» на вопросы, которые может задать пользователь относительно элементов сети. Т.е. данные и связи между ними хранятся в семантическом графе, а семантическая сеть имеет доступ к этому графу и оперирует с элементами графа.

Также класс SemanticNetwork содержит необходимые для перевода идентификаторов сети в нормальный вид константы.

public string Question(int conceptId1, int relationId, int conceptId2, bool fullDescription = false)

{

if (this.Graph.GetConceptById(conceptId1) == null && conceptId1 != SemanticNetwork.anyValue)

{

return String.Format("Не правильный запрос:\n - понятия с идентификатором {0} нет в базе\n", conceptId1);

}

if (this.Graph.GetConceptById(conceptId2) == null && conceptId2 != SemanticNetwork.anyValue)

{

return String.Format("Не правильный запрос:\n - понятия с идентификатором {0} нет в базе\n", conceptId2);

}

if (!this.Graph.RelationsDefinitions.Exists(rd => rd.Id == relationId) && relationId != SemanticNetwork.anyValue)

{

return String.Format("Не правильный запрос:\n - связи с идентификатором {0} нет в базе\n", relationId);

}

StringBuilder sb = new StringBuilder();

if (conceptId1 == SemanticNetwork.anyValue)

{

this.Graph.GetVertices().OfType<ConceptVertex>().ToList().ForEach(delegate(ConceptVertex c)

{

string s = this.Question(c.Id, relationId, conceptId2, fullDescription);

if (s != SemanticNetwork.error\_message)

sb.Append(s);

});

}

else

{

var c1 = this.Graph.GetConceptById(conceptId1);

if (conceptId2 == SemanticNetwork.anyValue)

{

this.Graph.GetVertices().OfType<ConceptVertex>().ToList().ForEach(delegate(ConceptVertex c)

{

if (c.Id != conceptId1)

{

string s = this.Question(conceptId1, relationId, c.Id, fullDescription);

if (s != SemanticNetwork.error\_message)

if (fullDescription)

{

sb.AppendLine(s);

}

else

{

sb.Append(s);

}

}

});

}

else

{

var c2 = this.Graph.GetConceptById(conceptId2);

var relBetween = this.Graph.FindRelation(conceptId1, conceptId2);

if (relBetween != null && (relBetween.Id == relationId || relationId == SemanticNetwork.anyValue))

{

if (relationId == SemanticNetwork.anyValue || fullDescription == false)

sb.AppendFormat("[{0}] {1} [{2}]\n", c1.Name, relBetween.Name, c2.Name);

else

sb.Append(SemanticNetwork.yes\_message);

}

else

{

if(fullDescription)

sb.Append(SemanticNetwork.error\_message);

}

}

}

if (sb.Length != 0)

{

return sb.ToString();

}

return SemanticNetwork.error\_message;

}

Имея в арсенале написанную программу для редактирования графа, я использовал её для визуализации семантического графа на плоскость, а также для выполнения некоторых манипуляций. Добавил возможность добавлять типы связи, а также добавил на панельку инструменты для выполнения запросов к семантической сети. Сохранение и открытие оставил прежним – простая сериализация графа и сети, а также добавил загрузку из текстового файла, что было описано в тексте задания к лабораторной работе.

В протоколах проведенных экспериментов я привел только те эксперименты, которые не выходят за рамки поставленной задачи, кроме разве что ввода с клавиатуры запроса. В моей программе это выполняется с помощью визуальных компонентов, что более наглядно и удобно. Возможность ввода запроса в виде *< код объекта1 | ? >:< код связи | ? >:< код объекта2 | ? >* я оставил пользователю посредством текстового файла, в котором я добавил 4-ый блок. В этом блоке можно написать несколько запросов в описанном выше формате, после открытия файла в программе ответы на все запросы из 4 блока будут выведены на экран.

# Протоколы проведенных экспериментов

Файл:

#1

10:самолет

11:летательный аппарат

12:птица

13:крылья

14:ласточка

#2

1:имеет часть:2

2:является:1

3:вес:0

#3

10:2:11

10:1:13

12:1:13

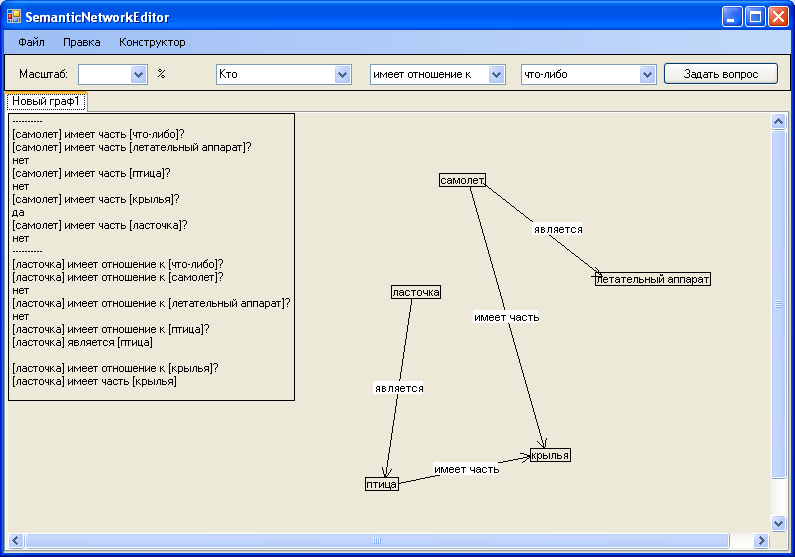
14:2:12

#4

10:1:?

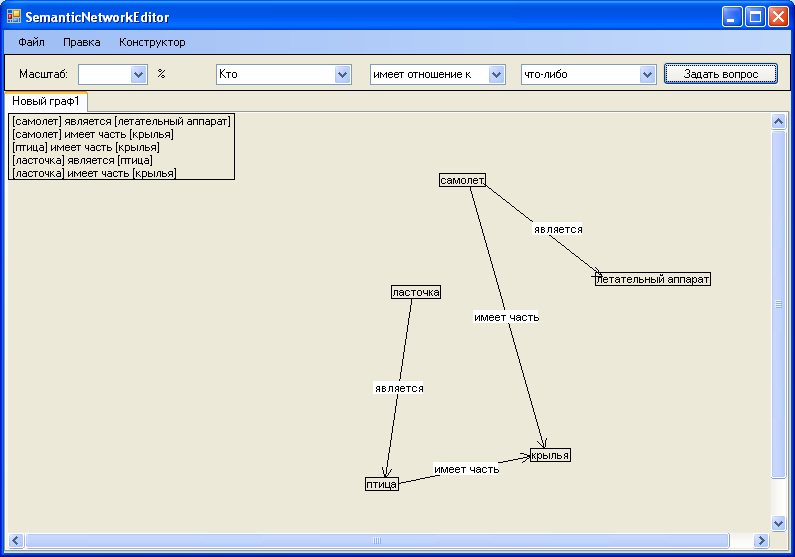
14:?:?

Результат:

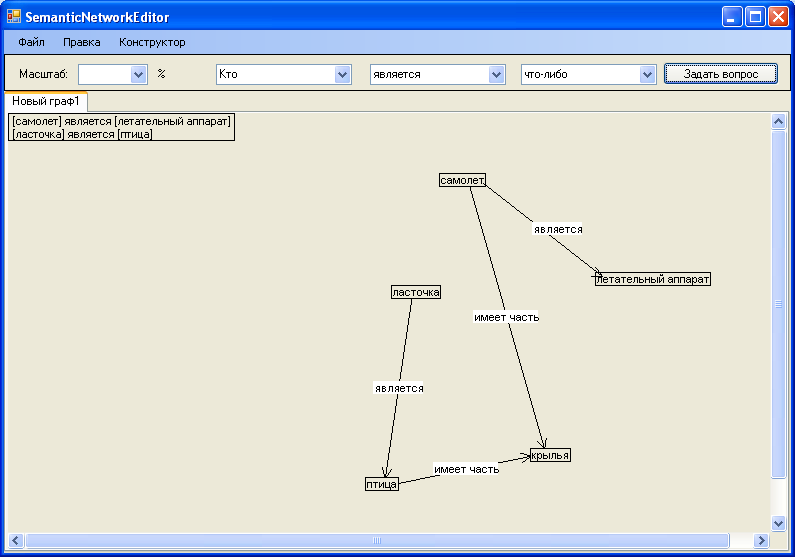


При открытии файла все ответы на запросы из 4 блока выводятся в «длинной» форме, со всеми вопросами и ответами на них. Если «задаваться вопросы» с помощью кнопки задать вопросы, ответы будут выглядеть так:

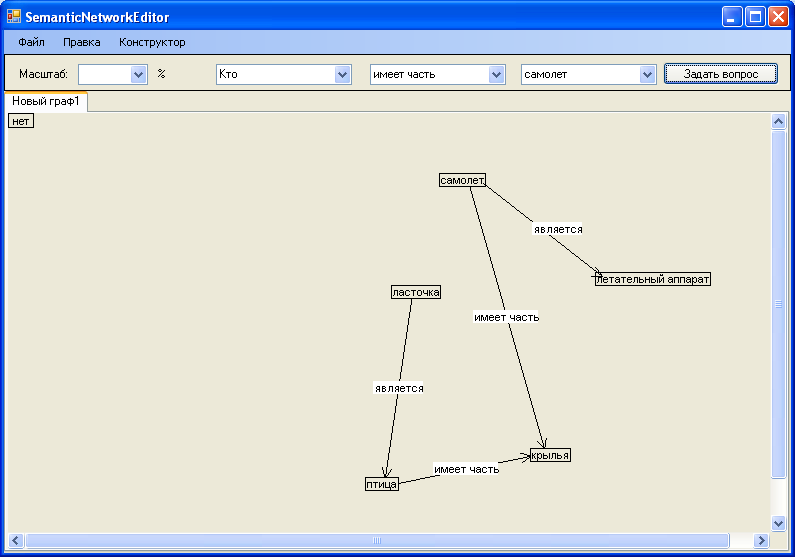
?:?:?



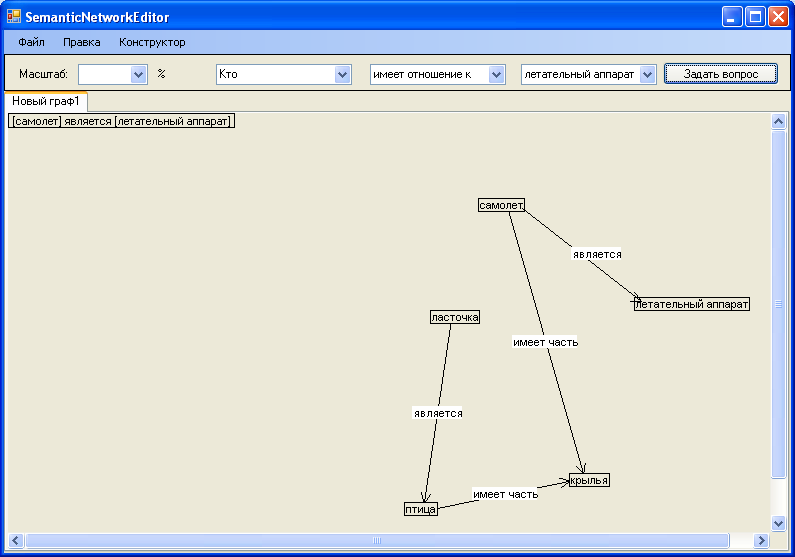
?:2:?



?:1:10



?:?:11



10:1:13

