МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное бюджетное образовательное учреждение высшего профессионального образования
«ТОМСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ СИСТЕМ УПРАВЛЕНИЯ И РАДИОЭЛЕКТРОНИКИ» (ТУСУР)

Кафедра Автоматизации обработки информации (АОИ)

Утверждаю:
Зав. каф. АОИ
_____Ю.П. Ехлаков

# ЭКСПЕРТНЫЕ СИСТЕМЫ

## Методические указания

к практическим занятиям и самостоятельным работам
для студентов направления 080700.62 — «Бизнес-информатика»

Разработчики:
профессор кафедры КИБЭВС
_____ И.А. Ходашинский
аспирант кафедры КИБЭВС
_____ А. В. Ахаев

Томск - 2012

# СОДЕРЖАНИЕ

# ВВЕДЕНИЕ

Практические и самостоятельные работы по дисциплине преследуют следующие цели:

– обобщение, систематизация, углубление, закрепление полученных теоретических знаний и основных положений искусственного интеллекта;

– обучение построению интеллектуальных систем;

– формирование умений применять полученные знания на практике.

**Практические занятия, их содержание (32 часа)**

| | | |
|---|---|---|
| 1. | Символизация естественного языка средствами логики высказываний | 2 |
| 2. | Логический вывод в логике высказываний | 4 |
| 3. | Логика предикатов первого порядка. Интерпретация. Нормальные формы | 4 |
| 4. | Выводы в логических моделях первого порядка | 6 |
| 5. | Знакомство с Visual Prolog. Обработка списков | 4 |
| 6. | Программирование баз знаний в среде Visual Prolog | 4 |
| 7. | Сортировка | 4 |
| 8. | Представление графов и поиск пути на графе | 4 |
| | Итого | 32 |

**Самостоятельная работа (60 часов)**

Самостоятельная работа включает следующие задания:

| № | Наименование работы | Часов | Контроль |
|---|---|---|---|
| 1 | Подготовка к лекциям, повторение учебного материала предыдущих лекций | 6 | Тестовый опрос |
| 2 | Подготовка к практическим занятиям, повторение изучения лекционного материала (проверка – на практических занятиях) | 8 | Проверка на занятиях |
| 3 | Выполнение домашних заданий | | |
| 3.1 | Символизация естественного языка средствами логики высказываний | 1 | Опрос, тест |
| 3.2 | Логический вывод в логике высказываний | 1 | Опрос, тест |
| 3.3 | Логика предикатов первого порядка. Интерпретация. Нормальные формы | 1 | Опрос, тест |
| 3.4 | Выводы в логических моделях первого порядка | 1 | Опрос, тест |
| 3.5 | Знакомство с Visual Prolog. Обработка списков | 1 | Опрос, тест |

| | | | |
|---|---|---|---|
| 3.6 | Программирование баз знаний в среде Visual Prolog | 1 | Опрос, тест |
| 3.7 | Сортировка | 1 | Опрос, тест |
| 3.8 | Представление графов и поиск пути на графе | 1 | Опрос, тест |
| | Подготовка к экзамену | 36 | экзамен |
| | **Итого** | 60 | |

# ЗАНЯТИЕ №1. СИМВОЛИЗАЦИЯ ЕСТЕСТВЕННОГО ЯЗЫКА СРЕДСТВАМИ ЛОГИКИ ВЫСКАЗЫВАНИЙ

*Цель занятия* – освоить приемы трансляции предложений естественного языка в язык формул логики высказываний.

## *Основы*

В логике высказываний предполагается, что мир может быть описан элементарными предложениями или высказываниями и логическими связями между ними. Кроме этого допущения приняты еще два следующих:

- простому предложению или высказыванию можно приписать истинностное значение;
- сложные предложения, образуются путем видоизменения некоторого предложения с помощью слова "НЕ" ($\sim$ ) или путем связывания простых предложений с помощью слов "И"( $\wedge$), "ИЛИ"( $\vee$), "ЕСЛИ, ТО"($\rightarrow$), "ТОГДА И ТОЛЬКО ТОГДА, КОГДА"( $\leftrightarrow$). Эти пять слов называют сентенциональными, пропозициональными или логическими связками, каждая из них имеет свое название " $\sim$ " – отрицание, "$\wedge$" – конъюнкция, "$\vee$" – дизъюнкция, "$\rightarrow$" – импликация, "$\leftrightarrow$" – эквиваленция.

Таким образом, *высказывание* – это неразлагаемое и неанализируемое повествовательное предложение, которое может быть истинными или ложными, но не тем и другим одновременно. Высказывание, состоящее из одного предложения, называют простым или элементарным.

"Истинна" или "ложь", которая приписывается высказыванию, и есть *истинностное значение* высказывания. Для краткости "истинна" обозначается как И, а "ложь" – Л. Высказывания обозначаются заглавными буквами или цепочкой таких букв. Например, Козьма Прутков считает, что

**P**: "Военные люди защищают отечество",

**Q**: "Ветер есть дыхание природы",

**R**: "Новые сапоги всегда жмут" [1].

Символы **P**, **Q**, **R** и др., которые используются для обозначения элементарных высказываний, называются *атомарными формулами* или *атомами*.

Примеры сложных высказываний от Козьмы Пруткова: "Чиновник умирает, а его ордена остаются на лице земли ", "Хочешь быть красивым, поступи в гусары". Примем следующие обозначения:

**M**: чиновник умирает;

**L**: ордена чиновника остаются на лице земли;

**B**: хотеть быть красивым;

**G**: поступать в гусары.

Тогда два последних предложения могут быть записаны в виде формул как $\mathbf{M} \wedge \mathbf{L}$, $\mathbf{B} \rightarrow \mathbf{G}$, соответственно.

### *Символизация*

Рассмотрим некоторые аспекты символизации естественного языка и применение логических связок. Операция конъюнкции в логике высказываний и союз "и" в повседневной речи употребляются в одном и том же смысле. Однако в обыденной речи не принято соединять союзом "и" два далеких по смыслу предложения (*ироничное*: "в огороде бузина, а в Киеве дядька"). В то время как в логике высказываний операция конъюнкции соединяет два любых высказывания. Операции конъюнкции соответствуют следующие выражения:

- **A** и **B**;
- не только **A**, но и **B**;
- **B**, хотя и **A**;
- **A**, а также **B**;
- как **A**, так и **B**;
- **A** вместе с **B**.

В повседневной речи союз "или" употребляется в двух различных смыслах: исключающем и не исключающем, а операция дизъюнкции всегда употребляется в не исключающем смысле.

Употребление слов "если …, то …" в повседневной речи существенно отличается от применения в логике высказываний. В предложении "если **A**, то **B**" обыденной речи подразумевается, что **B** логически следует из **A**, в то время как в логике высказываний, не рассматривающей смысла предложений, этого не требуется. Кроме того ложность предложения **A** в повседневной речи влечет либо ложность **B**, либо потерю смысла всего предложения "если **A**, то **B**". Таким образом, трансляция естественно языкового предложения в предложение логики высказываний при кажущейся простоте таковой не является. Здесь требуется понять смысл предложения, а затем уже конструировать формулы логики высказываний. Операции импликации ($\mathbf{A} \rightarrow \mathbf{B}$) соответствуют следующие выражения естественного языка:

- **B**, если **A**;

- **A** влечет **B**;
- **A** является причиной **B**;
- **B** является следствием **A**;
- в случае **A** имеет место **B**;
- коль скоро **A**, то **B**;
- **B**, так как **A**;
- **B,** потому что **A**.

Операции эквиваленции (**A** ↔ **B**) соответствуют следующие выражения естественного языка:

- **A**, если и только если **B**;
- если **A**, то **B** и обратно;
- **A**, если **B,** и **B**, если **A**;
- **A** эквивалентно **B**;
- **A** равносильно **B**.

### *Формулы*

*Правильно построенные формулы* или просто *формулы* в логике высказываний определяются рекурсивно следующим образом:

- Атом есть формула.
- Если **G** – формула, то (~**G**) – формула.
- Если **G** и **H** – формулы, то (**G** ∧ **H**), (**G** ∨ **H**), (**G** → **H**) и (**G** ↔ **H**) – формулы.
- Других формул нет.

Логическим связкам приписан следующий убывающий ранг:

**↔, →, ∨ , ∧, ~,**

таким образом, связка с большим рангом имеет большую область действия. Формула

**P** ↔ ~**Q** ∨ **R** → **S** означает (**P** ↔ (((~**Q**) ∨ **R**) → **S**)).

Если **G**, **H** – две формулы, тогда истинность формул (~**G**) , (**G** ∧ **H**), (**G** ∨ **H**), (**G** → **H**), (**G** ↔ **H**) определяется по истинностным значениям атомов, входящих в эту формулу и следующей истинностной таблице.

*Истинностная таблица*

| **G** | **H** | ~**G** | **G** ∧ **H** | **G** ∨ **H** | **G** → **H** | **G** ↔ **H** |
|---|---|---|---|---|---|---|
| И | И | Л | И | И | И | И |
| И | Л | Л | Л | И | Л | Л |
| Л | И | И | Л | И | И | Л |
| Л | Л | И | Л | Л | И | И |

*Интерпретацией* формулы является такое приписывание истинностных значений атомам, входящим в формулу, при котором каждому из них приписано либо И, либо Л, но не оба одновременно. Таким образом, если истинностные значения атомов известны, то истинностное значение формулы может быть определено индуктивным способом в соответствии с приведенной выше истинностной таблицей. Если формула содержит $n$ различных атомов, то эта формула имеет $2^n$ интерпретаций. Ниже приведена истинностная таблица для формул $\mathbf{P \to (Q \to P)}$ и $\mathbf{(P \to Q) \to P}$

| **P** | **Q** | **P → Q** | **Q → P** | **P → (Q → P)** | **(P → Q) → P** |
|---|---|---|---|---|---|
| И | И | И | И | И | И |
| И | Л | Л | И | И | И |
| Л | И | И | Л | И | Л |
| Л | Л | И | И | И | Л |

Интерпретацию формулы, содержащей атомы $\mathbf{A_1, A_2, \ldots, A_n}$, удобно представлять в виде

$I = \{m_1, m_2, \ldots, m_n\}$,

где $\mathbf{m_j}$ есть $\mathbf{A_j}$ или $\mathbf{\sim A_j}$. Если $\mathbf{m_j}$ есть $\mathbf{A_j}$, то атому $\mathbf{A_j}$ присвоено значение **И**, в противном случае **Л**. Например, в интерпретации {**~P, Q**} формула $\mathbf{(P \to Q) \to P}$ "ложна", а в интерпретации {**P, ~Q**} эта же формула "истинна".

Формула истинная при всех возможных интерпретациях называется *общезначимой формулой* или *тавтологией*. Формула ложная при всех возможных интерпретациях называется *противоречивой* (или *невыполнимой*). Общезначимую и противоречивую формулу будем обозначать !, э, соответственно. Формула, которая не является общезначимой или противоречивой, называется *необщезначимой, непротиворечивой или выполнимой*.

Общезначимость и противоречивость формулы может быть определена с использованием таблицы истинности, но в силу экспоненциального роста размерности числа интерпретаций с ростом числа входящих в формулу атомов, такой метод не всегда является приемлемым. Такое положение привело к необходимости разработки правил преобразования формул. Преобразования выполняются путем замены в преобразуемой формуле некоторой ее части на подформулу, эквивалентную заменяемой. Две формулы *эквивалентны*, если их истинностные значения совпадают при всех интерпретациях. Например, формула $\mathbf{(P \to Q) \to P}$ эквивалентна формуле **P**. Для ведения преобразований необходимо иметь минимальный запас эквивалентных формул, ниже

приведены десять законов преобразования, здесь F, G и H являются формулами, символ "=" это знак эквивалентности.

| | | |
|---|---|---|
| 1. | $\mathbf{F} \leftrightarrow \mathbf{G} = (\mathbf{F} \rightarrow \mathbf{G}) \wedge (\mathbf{G} \rightarrow \mathbf{F})$; | |
| 2. | $\mathbf{F} \rightarrow \mathbf{G} = \sim\mathbf{F} \vee \mathbf{G}$; | |
| 3. | $\mathbf{F} \vee \mathbf{G} = \mathbf{G} \vee \mathbf{F}$; | $\mathbf{F} \wedge \mathbf{G} = \mathbf{G} \wedge \mathbf{F}$; |
| 4. | $(\mathbf{F} \vee \mathbf{G}) \vee \mathbf{H} = \mathbf{F} \vee (\mathbf{G} \vee \mathbf{H})$; | $(\mathbf{F} \wedge \mathbf{G}) \wedge \mathbf{H} = \mathbf{F} \wedge (\mathbf{G} \wedge \mathbf{H})$; |
| 5. | $\mathbf{F} \vee (\mathbf{G} \wedge \mathbf{H}) = (\mathbf{F} \vee \mathbf{G}) \wedge (\mathbf{F} \vee \mathbf{H})$; | $\mathbf{F} \wedge (\mathbf{G} \vee \mathbf{H}) = (\mathbf{F} \wedge \mathbf{G}) \vee (\mathbf{F} \wedge \mathbf{H})$; |
| 6. | $\mathbf{F} \vee э = \mathbf{F}$; | $\mathbf{F} \wedge ! = \mathbf{F}$; |
| 7. | $\mathbf{F} \vee \mathbf{F} = \mathbf{F}$; | $\mathbf{F} \wedge \mathbf{F} = \mathbf{F}$; |
| 8. | $\mathbf{F} \vee ! = !$; | $\mathbf{F} \wedge э = э$; |
| 9. | $\mathbf{F} \vee \sim\mathbf{F} = !$; | $\mathbf{F} \wedge \sim\mathbf{F} = э$; |
| 10. | $\sim(\sim\mathbf{F}) = \mathbf{F}$; | |
| 11. | $\sim(\mathbf{F} \vee \mathbf{G}) = \sim\mathbf{F} \wedge \sim\mathbf{G}$; | $\sim(\mathbf{F} \wedge \mathbf{G}) = \sim\mathbf{F} \vee \sim\mathbf{G}$. |

Правила преобразования под номером три называются коммутативными законами; законы под номером четыре – ассоциативными законами; законы под номером пять – дистрибутивными законами; семь – закон идемпотентности; девять – законы дополнения; десять – закон двойного отрицания; одиннадцать – законами де Моргана.

В логике высказываний определены две нормальные формы: дизъюнктивная и конъюнктивная. Формула находится в *дизъюнктивной нормальной форме*, если она имеет следующий вид: $\mathbf{F_1} \vee \mathbf{F_2} \vee \ldots \vee \mathbf{F_n}$, где каждая подформула $\mathbf{F_i}$ – это конъюнкция атомов или отрицания атомов. Формула находится в *конъюнктивной нормальной форме*, если она имеет следующий вид: $\mathbf{F_1} \wedge \mathbf{F_2} \wedge \ldots \wedge \mathbf{F_n}$, где каждая подформула $\mathbf{F_i}$ – это дизъюнкция атомов или отрицания атомов. *Литера* – это атом или отрицание атома. *Дизъюнкт* – это дизъюнкция литер. *Единичный дизъюнкт* – это дизъюнкт, состоящий из одной литеры. Дизъюнкт, не содержащий никаких литер, называется *пустым дизъюнктом*. Формула, находящаяся в конъюнктивной нормальной форме, может быть представлена как множество входящих в форму дизъюнктов. Например, формула $(\mathbf{P} \vee \mathbf{Q}) \wedge (\mathbf{P} \vee \mathbf{R} \vee \sim\mathbf{Q}) \wedge (\sim\mathbf{Q} \vee \mathbf{P}) \wedge \sim\mathbf{P}$ может быть представлена как множество $\{\mathbf{P} \vee \mathbf{Q}, \mathbf{P} \vee \mathbf{R} \vee \sim\mathbf{Q}, \sim\mathbf{Q} \vee \mathbf{P}, \sim\mathbf{P}\}$.

## *Задание*

I. Определить, верно ли, что формула общезначима, необщезначима, противоречива, непротиворечива или обладает некоторой комбинацией этих свойств:

1) $\sim(\sim P)) \to P$ 2) $P \to (P \wedge Q)$
3) $\sim(P \vee Q) \vee \sim Q$ 4) $(P \vee Q) \to P$
5) $(P \to Q) \to (\sim Q \to \sim P)$ 6) $(P \to Q) \to (Q \to P)$

II. Преобразуйте приведенные в задании I формулы в дизъюнктивную и конъюнктивную нормальные формы.

III. Записать следующие предложения с помощью формул.

1) Если влажность высока, то либо после полудня, либо вечером будет дождь.

2) Лечение от рака не будет найдено, пока не определены его причины и не найдены новые лекарства.

3) Данное отношение есть отношение эквивалентности тогда и только тогда, когда оно рефлексивно, симметрично и транзитивно.

IV. Имеются следующие утверждения:

P: ему нужен доктор; Q: ему нужен адвокат;
R: с ним произошел несчастный случай;
S: он болен; U: он ранен.

Записать следующие формулы на естественном языке:

$(S \to P) \wedge (R \to Q)$
$P \to (S \vee U)$
$(P \wedge Q) \to R$
$(P \wedge Q) \leftrightarrow (S \wedge U)$
$\sim(S \vee U) \to \sim P$

### *Методические указания*

Ознакомьтесь с основными конструкциями логики высказываний. Уясните правила построения формул в логике высказываний.

Обратите внимание на сходство и отличия в применении логических связок в естественном языке и в логических формулах. Изучите бесскобочную форму записи логических формул.

Раскройте содержание понятия «интерпретация формулы логики высказываний», а также понятий «общезначимость», «противоречивость», «необщезначимость», «непротиворечивость», «выполнимость» формул логики высказываний. Изучите эти понятия на своих собственных примерах.

Ознакомьтесь с правилами эквивалентных преобразований формул. Обратите внимание на то, что эквивалентные преобразования действуют в обоих направлениях.

## *Самостоятельная работа*

I. Определить, верно ли, что формула общезначима, необщезначима, противоречива, непротиворечива или обладает некоторой комбинацией этих свойств:

1) $P \vee (P \to Q)$
2) $P \vee (Q \to \sim P)$
3) $(P \wedge (Q \to P)) \to P$
4) $(P \vee \sim Q) \wedge (\sim P \vee Q)$

II. Преобразуйте приведенные в задании I формулы в дизъюнктивную и конъюнктивную нормальные формы.

III. Записать следующие предложения с помощью формул.

1) Если рабочие или администрация упорствуют, то забастовка будет урегулирована тогда и только тогда, когда правительство добьется ее судебного запрещения, но войска не будут посланы на завод.

2) Если парламент отказывается принять новые законы, то забастовка не будет окончена, если она только не длится более года и президент не уходит в отставку; либо парламент примет новые законы, либо забастовка не окончится, хотя и продолжается более года.

IV. Даны следующие высказывания:

A: Иванов купил компьютер.

B: Петров успешно сдал экзамен.

C: Сидоров уехал в другой город.

Переведите на естественный язык следующие формулы логики высказываний:

$\sim A \to B$;
$\sim(\sim A \to B)$;
$\sim(\sim A \to B) \to C$;
$(\sim A \wedge B) \to C$ ;
$\sim(B \wedge C) \to \sim A$ ;
$\sim(\sim B \wedge \sim C) \to A$ ;

V. Приведите свои примеры сложных высказываний на естественном языке и переведите их в логические формулы.

## *Контрольные вопросы*

1. Какие допущения приняты при описании мира с помощью логических моделей?
2. Что такое правильно построенная формула в логике высказываний?

## ЗАНЯТИЕ №2. ЛОГИЧЕСКИЙ ВЫВОД В ЛОГИКЕ ВЫСКАЗЫВАНИЙ

*Цель занятия* – освоить приемы и способы логического вывода в логике высказываний.

### *Способы вывода в логических моделях нулевого порядка*

Задача логики – описать мир в виде логических формул и дать теорию вывода. Практически теория выводов сводится к получению критериев и алгоритмов для решения вопроса о том, можно ли некоторую цепочку рассуждений, основываясь на ее форме, считать правильной. Цепочка рассуждений представляет собой конечную последовательность высказываний, приводимых в обоснование утверждения того, что последнее высказывание в этой последовательности (заключение) может быть выведено из некоторых начальных высказываний (посылок). Это приводит к понятию "логического следствия".

*ОПРЕДЕЛЕНИЕ*. Пусть даны формулы $\mathbf{F_1}$, $\mathbf{F_2}$, ..., $\mathbf{F_n}$ и формула **G**. Формула **G** есть *логическое следствие* формул $\mathbf{F_1}$, $\mathbf{F_2}$, ..., $\mathbf{F_n}$, если для всякой интерпретации, в которой формула $\mathbf{F_1} \wedge \mathbf{F_2} \wedge ... \wedge \mathbf{F_n}$ истинна, **G** также истинна. Формулы $\mathbf{F_1}$, $\mathbf{F_2}$, ..., $\mathbf{F_n}$ называются посылками, **G** – заключением.

*ТЕОРЕМА* 1. Пусть даны формулы $\mathbf{F_1}$, $\mathbf{F_2}$, ..., $\mathbf{F_n}$ и формула **G**. Тогда **G** есть логическое следствие формул $\mathbf{F_1}$, $\mathbf{F_2}$, ..., $\mathbf{F_n}$, если формула $((\mathbf{F_1} \wedge \mathbf{F_2} \wedge ... \wedge \mathbf{F_n}) \rightarrow \mathbf{G})$ общезначима.

*ТЕОРЕМА* 2. Пусть даны формулы $\mathbf{F_1}$, $\mathbf{F_2}$, ..., $\mathbf{F_n}$ и формула **G**. Тогда **G** есть логическое следствие формул $\mathbf{F_1}$, $\mathbf{F_2}$, ..., $\mathbf{F_n}$, если формула $(\mathbf{F_1} \wedge \mathbf{F_2} \wedge ... \wedge \mathbf{F_n} \wedge \sim\mathbf{G})$ противоречива.

Таким образом, вопрос о том, какие высказывания представляют собой логические следствия других высказываний, сводится к вопросу о том, какие высказывания общезначимы или противоречивы. Это, в свою очередь, дает возможность превратить *ОПРЕДЕЛЕНИЕ* и *ТЕОРЕМЫ* 1 и 2 в рабочий аппарат для логического вывода. Ниже приведены пять способов логического вывода в логике высказываний. В качестве примера рассматриваются следующие формулы: $F_1$: **P**; $F_2$: **R**; $F_3$: $\mathbf{Q} \wedge \mathbf{R} \rightarrow \sim\mathbf{R}$; G: $\sim \mathbf{Q}$.

*Способ* 1, *вычисление истинностного значения*. Здесь вывод основан на определении логического следствия и на истинностных таблицах.

| **P** | **R** | **Q** | **Q ∧ R** | **~R** | **Q ∧ R → ~R** | **P ∧ R ∧ ((Q ∧ R) → ~R)** | **~ Q** |
|---|---|---|---|---|---|---|---|
| И | И | И | И | Л | Л | Л | Л |
| И | И | Л | Л | Л | И | И | И |
| И | Л | И | Л | И | И | Л | Л |
| И | Л | Л | Л | И | И | Л | И |
| Л | И | И | И | Л | Л | Л | Л |
| Л | И | Л | Л | Л | И | Л | И |
| Л | Л | И | Л | И | И | Л | Л |
| Л | Л | Л | Л | И | И | Л | И |

Из таблицы видно, что есть только одна интерпретация, где формула **P ∧ R ∧ ((Q ∧ R) → ~R)** является истинной, в этой же интерпретации формула **~Q** является истинной, следовательно, формула **~Q** является логическим следствием формул **P**, **R**, **Q ∧ R → ~R**.

*Способ таблиц истинности* 2. Здесь в качестве аппарата для логического вывода может быть использована ТЕОРЕМА 1 и метод истинностных таблиц.

| **P** | **R** | **Q** | **Q ∧ R** | **~R** | **Q ∧ R → ~R** | **P ∧ R ∧ ((Q ∧ R) → ~R)** | **~ Q** | **$F_1 \wedge F_2 \wedge F_3 \rightarrow G$** |
|---|---|---|---|---|---|---|---|---|
| И | И | И | И | Л | Л | Л | Л | И |
| И | И | Л | Л | Л | И | И | И | И |
| И | Л | И | Л | И | И | Л | Л | И |
| И | Л | Л | Л | И | И | Л | И | И |
| Л | И | И | И | Л | Л | Л | Л | И |
| Л | И | Л | Л | Л | И | Л | И | И |
| Л | Л | И | Л | И | И | Л | Л | И |
| Л | Л | Л | Л | И | И | Л | И | И |

Из таблицы видно, что есть формула **P ∧ R ∧ ((Q ∧ R) → ~R) → ~Q** является общезначимой, значит формула **~Q** является логическим следствием формул **P**, **R**, **Q ∧ R → ~R**.

*Способ таблиц истинности* 3. В качестве аппарата для логического вывода может быть использована ТЕОРЕМА 2 и метод истинностных таблиц.

| P | R | Q | $Q \wedge R$ | $\sim R$ | $Q \wedge R \rightarrow \sim R$ | $P \wedge R \wedge ((Q \wedge R) \rightarrow \sim R)$ | $F_1 \wedge F_2 \wedge F_3 \wedge \sim G$ |
|---|---|---|---|---|---|---|---|
| И | И | И | И | Л | Л | Л | Л |
| И | И | Л | Л | Л | И | И | Л |
| И | Л | И | Л | И | И | Л | Л |
| И | Л | Л | Л | И | И | Л | Л |
| Л | И | И | И | Л | Л | Л | Л |
| Л | И | Л | Л | Л | И | Л | Л |
| Л | Л | И | Л | И | И | Л | Л |
| Л | Л | Л | Л | И | И | Л | Л |

Из таблицы видно, что есть формула $P \wedge R \wedge ((Q \wedge R) \rightarrow \sim R) \wedge \sim Q$ является противоречивой, значит формула $\sim Q$ является логическим следствием формул $P$, $R$, $Q \wedge R \rightarrow \sim R$.

*Способ* 4*, алгебраический*, здесь в качестве аппарата для логического вывода используется ТЕОРЕМА 1, а для доказательства общезначимости формулы – десять законов эквивалентных преобразований.

$P \wedge R \wedge ((Q \wedge R) \rightarrow \sim R) \rightarrow \sim Q = \sim(P \wedge R \wedge ((Q \wedge R) \rightarrow \sim R)) \vee \sim Q =$
$\sim(P \wedge R \wedge (\sim(Q \wedge R) \vee \sim R)) \vee \sim Q = \sim P \vee \sim R \vee (Q \wedge R \wedge R) \vee \sim Q =$
$\sim P \vee \sim R \vee (Q \wedge R) \vee \sim Q = \sim P \vee \sim R \vee (Q \wedge \sim Q) \vee (R \vee \sim Q) =$
$\sim P \vee \sim R \vee R \vee \sim Q = \sim P \vee ! \vee \sim Q = !.$

*Способ* 5*,* также *алгебраический*, только здесь использована ТЕОРЕМА 2, а для доказательства противоречивости формулы – десять законов эквивалентных преобразований.

$P \wedge R \wedge ((Q \wedge R) \rightarrow \sim R) \wedge Q = P \wedge R \wedge (\sim(Q \wedge R) \vee \sim R) \wedge Q =$
$P \wedge R \wedge (\sim Q \vee \sim R \vee \sim R) \wedge Q = P \wedge R \wedge (\sim Q \vee \sim R) \wedge Q =$
$P \wedge R \wedge ((\sim Q \wedge Q) \vee (\sim R \wedge \sim Q)) = P \wedge R \wedge \sim R \wedge \sim Q =$
$P \wedge э \wedge \sim Q = э.$

*Способ* 6*, алгоритм редукции*. Этот способ позволяет доказывать общезначимость формул приведением их к абсурду. Этот способ удобен, когда формула содержит много импликаций. Пусть посылка $(P \wedge Q) \rightarrow R$, а заключение $P \rightarrow (Q \rightarrow R)$, тогда заключение является логическим следствием посылки, если формула $((P \wedge Q) \rightarrow R) \rightarrow (P \rightarrow (Q \rightarrow R))$ является общезначимой. Предположим, что в некоторой интерпретации эта формула ложна. Это возможно только тогда, когда формула $(P \wedge Q) \rightarrow R$ истинна, а формула $P \rightarrow (Q \rightarrow R)$ ложна. Формула $P \rightarrow (Q \rightarrow R)$ ложна только тогда, когда $P$ = **И**, $Q$ = **И**, $R$ = **Л**, что противоречит предположению $(P \wedge Q) \rightarrow R$ = **И**, следовательно, формула $((P \wedge Q) \rightarrow R) \rightarrow (P \rightarrow (Q \rightarrow R))$ является общезначимой.

*Способ* 7, *алгоритм Девиса и Патнема*. Данный алгоритм основан на использовании конъюнктивной нормальной форме, представленной как множество дизъюнктов **S**. Алгоритм состоит из следующих четырех правил:

1. Правило тавтологии. Из множества дизъюнктов **S** вычеркиваются все тавтологичные дизъюнкты.
2. Правило однолитерных дизъюнктов. Если в множестве **S** есть единичный дизъюнкт **C**, то, вычеркнув из множества **S** дизъюнкт **C**, получим множество **S'**. Если **S'** пусто, то **S** выполнимо; иначе, вычеркнув из множества **S'** дизъюнкт ~**C**, получим множество **S''**. Множество **S** невыполнимо, когда невыполнимо **S''**. Если ~**C** является единичным дизъюнктом, то при его вычеркивании он превращается в противоречивую формулу э.
3. Правило чистых литер. Литера **L** в дизъюнкте из **S** называется чистой, если ~**L** не появляется ни в каком дизъюнкте из **S**. Если литера **L** чистая в **S**, то, вычеркнув из множества **S** все дизъюнкты, содержащие **L**, получим множество **S'**. Множество **S** невыполнимо, когда невыполнимо **S'**.
4. Правило расщепления. Если множества дизъюнктов **S** может быть представлено следующим образом:
   $(A_1 \vee L) \wedge \ldots \wedge (A_n \vee L) \wedge (A_1 \vee \sim L) \wedge \ldots \wedge (B_m \vee \sim L) \wedge R$,
   где $A_i$, $B_j$, **R** не содержат литер **L** и ~**L**, то получим два множества расщепления $S' = A_1 \wedge \ldots \wedge A_n \wedge R$ и $S'' = B_1 \wedge \ldots \wedge B_m \wedge R$. Множество **S** невыполнимо, когда невыполнимы множества **S'** и **S''**.

Используя алгоритм Девиса и Патнема, покажем, что формула $(P \vee Q) \wedge (P \vee R) \wedge (\sim Q \vee \sim R) \wedge \sim P \wedge U$ невыполнима.

**S** = { **P** ∨ **Q** , **P** ∨ **R**, ~**Q** ∨ ~**R**, ~**P**, **U** }

{ **Q**, **R**, ~**Q** ∨ ~**R**, **U** } правило 2, единичный дизъюнкт **C** = ~**P**;

{ **R**, ~**R**, **U** } правило 2, единичный дизъюнкт **C** = **Q**;

{э, **U** } правило 2, единичный дизъюнкт **C** = **R**.

Так как множество содержит невыполнимый дизъюнкт, то оно невыполнимо.

*Способ 8, принцип резолюций.* Данный метод является обобщением правила однолитерных дизъюнктов Девиса и Патнема и также основан на использовании конъюнктивной нормальной форме, представленной как множество дизъюнктов **S**. Обобщение правила связано с тем, что оно применяется к любой паре дизъюнктов, не обязательно единичных. Правило это называется правилом резолюций и формулируется следующим образом:

📖 **Если в дизъюнкте $C_1$ существует литера L, а в дизъюнкте $C_2$ существует литера ~L, то, вычеркнув литеры L и ~L из $C_1$ и $C_2$, соответственно, построим дизъюнкцию оставшихся дизъюнктов, которая называется *резольвентой* дизъюнктов $C_1$ и $C_2$.**

Рассмотрим, например, дизъюнкты $\mathbf{C_1}$: **P** ∨ **Q**, $C_2$: **~Q** ∨ **~R** ∨ **U**, резольвентой дизъюнктов $\mathbf{C_1}$ и $\mathbf{C_2}$ будет следующий дизъюнкт: **P** ∨ **~R** ∨ **U**.

📖 **Важное свойство резольвенты. Любая резольвента дизъюнктов $C_1$ и $C_2$ является логическим следствием дизъюнктов $C_1$ и $C_2$. Для невыполнимого множества дизъюнктов применением правила резолюций можно получить пустой дизъюнкт э.**

Метод резолюций основан на проверке того, содержит ли исходное множество дизъюнктов пустой дизъюнкт, если множество содержит пустой дизъюнкт, то это множество невыполнимо, в противном случае проверяется, можно ли получить пустой дизъюнкт из исходного множества. Такой процесс проверки называется выводом.

*Выводом* из конечного множества дизъюнктов **S** называется конечная последовательность $\mathbf{C_1}$, $\mathbf{C_2}$, **…,** $\mathbf{C_n}$ дизъюнктов**,** всякий элемент $\mathbf{C_i}$ которой или принадлежит множеству **S**, или является резольвентой дизъюнктов, предшествующих данному элементу $\mathbf{C_i}$**.** Вывод пустого дизъюнкта э из **S** называется опровержением или доказательством невыполнимости **S**.

Используя метод резолюций, проведем вывод и покажем, что формула **(P** ∨ **Q)** ∧ **(P** ∨ **R)** ∧ **(~Q** ∨ **~R)** ∧ **~P** ∧ **U** невыполнима. Запишем множество дизъюнктов **S** = **{ P** ∨ **Q , P** ∨ **R, ~Q** ∨ **~R, ~P, U }** следующим образом:

1. **P** ∨ **Q**
2. **P** ∨ **R**
3. **~Q** ∨ **~R**
4. **~P**
5. **U.**

Из 2) и 3) получим резольвенту

6. **P** ∨ **~Q.**

Из 1) и 6) получим резольвенту

7. **P.**

Из 4) и 7) получим резольвенту

8. э**.**

Так как э есть логическое следствие множество дизъюнктов **S**, то **S** невыполнимо.

📖 **Если возможно описать задачу в терминах логики высказываний, то применив любой из указанных восьми способов вывода, можно, доказав противоречивость или общезначимость формулы, решить поставленную задачу.**

### *Задание*

I. Докажите всеми возможными способами, что (~Q → ~P) есть логическое следствие (P → Q).

II. Или Маша и Ваня одного возраста, или Маша старше Вани. Если Маша и Ваня одного возраста, то Наташа и Ваня не одного возраста. Если Маша старше Вани, то Ваня старше Пети. Следовательно, или Наташа и Ваня не одного возраста, или Ваня старше Пети. Докажите.

III. Если завтра будет холодно, я надену шубу, если рукав будет починен. Завтра будет холодно, а, рукав не будет починен. Следовательно, я не надену шубу. Докажите.

### *Методические указания*

Разберитесь с понятием логического следствия. Выясните, как связаны между собой логическое следствие, общезначимость и противоречивость.

Изучите способы логического вывода в логике высказываний с точки зрения реализации их на компьютере.

### *Самостоятельная работа*

I. Докажите всеми возможными способами, что (P → Q) есть логическое следствие (~Q → ~P).

II. Если я поеду автобусом, а автобус опоздает, то я пропущу назначенное свидание. Если я пропущу назначенное свидание и буду огорчен, то мне не следует ехать домой. Если я не получу эту работу, то я буду огорчен и мне следует поехать домой. Следовательно, если я поеду домой автобусом, и автобус опоздает, то я получу эту работу. Докажите.

III. Если Степан не знал о необходимости декларировать доход, то он плохой законодатель. Если он знал и не декларировал, то он мошенник. Если Степан является плохим законодателем или мошенником, то ему нет места в Думе. Степан не декларировал свой доход. Следовательно, ему нет места в Думе. Доказать всеми возможными способами.

### *Контрольные вопросы*

1. Что такое логическое следствие?
2. Как можно превратить определение логического следствия и *ТЕОРЕМЫ* 1 и 2 в рабочий аппарат для логического вывода?

# ЗАНЯТИЕ №3. ЛОГИКА ПРЕДИКАТОВ ПЕРВОГО ПОРЯДКА. ИНТЕРПРЕТАЦИЯ. НОРМАЛЬНЫЕ ФОРМЫ

*Цель занятия* – освоить приемы трансляции предложений естественного языка в язык формул логики предикатов первого порядка и поиск интерпретации формул логики предикатов.

## *Основы*

В логике предикатов используется квантификация символически представляемая, как ∀, и называемая квантором всеобщности (общности). Запись **(∀x)** читается как «для всех **x**», «для всякого **x**», «для каждого **x**». Кроме квантора всеобщности, используется квантор существования, символически записываемый в виде **(∃x)**, что означает «существует **x**», «для некоторых **x**», «по крайней мере, для одного **x**».

Константы и переменные образуют более общее понятие — *терм*, который определяется следующим образом:

- константа — это терм;
- переменная — это терм;
- если $f$ — $n$-местный функциональный символ и $t_1, t_2, \ldots, t_n$ — термы, то $f(t_1, t_2, \ldots, t_n)$ — терм;
- других термов нет.

Синонимом для сложного терма вида $f(t_1, t_2, \ldots, t_n)$ будет «функция». Функция есть отображение списка констант в константу. Предикат — это тоже отображение списка констант, но не в константу, а в элемент множества {И, Л}.

📖 ***Основными символьными конструкциями логики предикатов являются константы, переменные, термы, предикатные символы, логические связки, кванторы.***

Если $P$ — $n$-местный предикатный символ и $t_1, t_2, \ldots, t_n$ — термы, то $P(t_1, t_2, \ldots, t_n)$ — *атом*. Понятие *формулы* рекурсивно определяется следующим образом:

- атом — это формула;
- если **G** и **H** — формулы, то (~**G**), (**G** ∧ **H**), (**G** ∨ **H**), (**G** → **H**) и (**G** ↔ **H**) — формулы.
- если **G** — формула и **x** — переменная, то **(∀x)G**, **(∃x)G** — формулы.
- других формул нет.

Здесь, как и в логике высказываний, круглые скобки могут быть опущены в соответствии с принятыми соглашениями о ранге логических связок, считая ранг кванторов наименьшим.

Примем следующие соглашения по использованию мнемонических имен, которые будут действовать по умолчанию:

- константы обозначаются строчными буквами *a, b, c, d, e* возможно с индексами;
- переменные — это обычно строчные буквы *x, y, z, u, v, w*, возможно с индексами;
- функциональные символы представляются буквами *f, g, h* или словами из строчных букв;
- предикатные символы обозначаются прописными буквами P, Q, R, S или словами из прописных букв.

### *Символизация*

Рассмотрим несколько примеров *формализации предложений естественного языка*. Пусть **R(x)** обозначает «**x** есть рациональное число», а **Q(x)** — «**x** есть действительное число», тогда предложение «каждое рациональное число есть число действительное» можно переформулировать в такое: «для любого x, если x есть рациональное число, то x есть действительное число» и символически представить в следующем виде: $(\forall x)(R(x) \rightarrow Q(x))$.

Смысл этого предложения сводится к тому, что множество рациональных чисел является подмножеством множества действительных чисел, что записывается как $\mathbf{R} \subseteq \mathbf{Q}$. Такая запись является типичной для высказываний «всякое нечто есть то-то».

Теперь рассмотрим утверждение «некоторые действительные числа являются рациональными». Формула, правильно отражающая это предложение, будет иметь вид: $(\exists x)(Q(x) \wedge R(x))$.

Это означает, что пересечение двух множеств действительных и рациональных чисел не является пустым $\mathbf{R} \cap \mathbf{Q} \neq \varnothing$. Типичная ошибка при формализации такого типа предложений заключается в том, что, исходя из правильности формализации предложений типа «всякое нечто есть то-то» в виде $(\forall x)(R(x) \rightarrow Q(x))$, делается ошибочное предположение в правильности формализации предложения типа «некоторое нечто есть то-то» формулой $(\exists x)(Q(x) \rightarrow R(x))$.

И еще одно предложение ошибочно — «не каждое действительное число есть число рациональное», что может быть прочтено как «не верно, что каждое действительное число является рациональным», формальная запись которого выглядит так: $\neg((\forall x)(Q(x) \rightarrow R(x)))$.

Переформулировав это предложение еще и как «существуют числа, которые являются действительными, но не являются рациональными», можно это же предложение представить и в виде такой формулы: $(\exists x)(Q(x) \wedge \sim R(x))$.

Таким образом, имеется две формулы, отражающие одно и то же предложение естественного языка, однако, в дальнейшем станет ясно, что эти формулы являются эквивалентными.

📖 ***Не существует простых синтаксических правил перевода предложений естественного языка в язык формул логики предикатов. Для перевода необходимо установить смысл предложения, а потом транслировать этот смысл в язык логики предикатов.***

## *Интерпретация*

Формулы логики предикатов могут быть интерпретированы, т. е. им может быть приписано истинностное значение. Однако наличие термов требует присвоения и им значения из предметной области. Таким образом, *интерпретация* формулы $F$ логики предикатов состоит из непустой предметной области $D$ и задания значений из данной предметной области всем константам, переменным и функциям, встречающимся в формуле $F$; каждому предикату, определенному на $D$, приписывается либо И, либо Л. Правила интерпретации для неквантифицированных и квантифицированных формул будут следующие:

- если истинностные значения формул **G** и **H** заданы, то истинностные значения формул $(\sim G)$, $(G \wedge H)$, $(G \vee H)$, $(G \rightarrow H)$ и $(G \leftrightarrow H)$ определяются по таблицам истинности;
- формула $(\forall x)G$ получит значение **И**, если значение **И** получит формула **G** для каждого **x** из области $D$, в противном случае формула **G** получит значение **Л**;
- формула $(\exists x)G$ получит значение **И**, если формула **G** получит значение **И** хотя бы для одного **x** из области $D$, в противном случае формула **G** получит значение **Л**.

Рассмотрим следующий пример: пусть дана формула

$$(\forall x)\,(\exists y)(P(x, a) \rightarrow Q(f(y))),$$

в следующей интерпретации: область $D = \{1,2\}$, константа $a = 1$, функция **f** принимает следующие значения:

| f(1) | f(2) |
|---|---|
| 2 | 1 |

оценка для предикатов следующая:

| **P(1, 1)** | **P(1, 2)** | **P(2, 1)** | **P(2, 2)** | **Q(1)** | **Q(2)** |
|---|---|---|---|---|---|
| **И** | **Л** | **И** | **Л** | **И** | **Л** |

Пусть **x** = 1, **y** = 1, определим истинностное значение формулы **P(x, a) → Q(f(y))** при данных значениях **x** и **y**.

**P(x, a) → Q(f(y)) = P(1, a) → Q(f(1)) = P(1, 1) → Q(2) = И → Л = Л**.

Положим **y** = 2.

**P(x, a) → Q(f(y)) = P(1, a) → Q(f(2)) = P(1, 1) → Q(1) = И → И = И**, т.е. для **x** = 1 существует такой **y** (а именно **y** = 2), что формула **P(1, a) → Q(f(y))** принимает значение **И**.

Пусть теперь **x** = 2, **y** = 1, определим истинностное значение формулы **P(x, a) → Q(f(y))** при данных значениях **x** и **y**.

**P(x, a) → Q(f(y)) = P(2, a) → Q(f(1)) = P(2, 1) → Q(2) = И → Л = Л**.

Положим **y** = 2

**P(x, a) → Q(f(y)) = P(2, a) → Q(f(2)) = P(2, 1) → Q(1) = И → И = И**.

Таким образом, и для **x** = 2 существует такой **y** (а именно **y** = 2), что формула **P(2, a) → Q(f(y))** принимает значение **И**. Так как для всех **x** из области *D* существует такое значение **y**, что формула **P(x, a) → Q(f(y))** истинна, то формула **(∀x) (∃y)(P(x, a) → Q(f(y)))** истинна в указанной интерпретации.

Формулы логики предикатов, также как и формулы логики высказываний, делятся на три класса: *общезначимые* — истинные во всех своих интерпретациях, *противоречивые* (*невыполнимые*) — ложные во всех своих интерпретациях, и *непротиворечивые* (*выполнимые*) — истинные только в отдельных своих интерпретациях.

### *Нормальные формы*

Понятие логического следствия формул, определенное для формул логики высказываний, справедливо и в логике предикатов; справедливы здесь и отношения между общезначимостью, противоречивостью и логическим следствием, заданные в виде *ТЕОРЕМ* 1 и 2. В общем случае язык логики высказываний является подмножеством языка логики предикатов первого порядка. Формула логики высказываний может рассматриваться как формула логики предикатов без кванторов, функций и переменных.

📖 ***Формула логики предикатов имеет в общем случае бесконечное число областей интерпретации или предметных областей и, как следствие, бесконечное число интерпретаций, а значит невозможно доказать противоречивость или общезначимость формулы перебором всех ее интерпретаций***.

Цель дальнейшего изложения — определить процедуру проверки противоречивости формул логики предикатов. Для решения этой задачи введены еще две нормальные формы: *предваренная и сколемовская стандартная*, помимо двух ранее определенных: конъюнктивной и дизъюнктивной. Введение нормальных форм позволит упростить алгоритмы поиска противоречивости формул логики предикатов.

*Предваренной (префиксной) нормальной формой* называется формула, состоящая из префикса и матрицы, здесь префикс — это конечная последовательность кванторных комплексов, а матрица — это формула, не содержащая кванторных комплексов, т.е. формула имеет следующий вид:

$$(Q_1x_1)\,(Q_2x_2)\,\ldots\,(Q_nx_n)\,M,$$

где $Q_i$ есть $\forall$ или $\exists$ для i = 1, 2, …, n.

Рассмотрим метод преобразования формулы логики предикатов в предваренную нормальную форму. Одиннадцать законов преобразования были определены ранее при рассмотрении логики высказываний (см. п. 2.1.3), эти законы действуют и в логике предикатов. Ниже рассмотрим дополнительные пары эквивалентных формул, содержащих кванторы (пусть **F** и **H** — это формулы, содержащие переменную **x**, **G** есть формула, которая не содержит переменной **x**, а **Q** — это $\forall$ или $\exists$):

12) $(Qx)F(x) \wedge G = (Qx)(F(x) \wedge G);$ $\quad (Qx)F(x) \vee G = (Qx)(F(x) \vee G);$

13) $\sim((\forall x)\,F\,(x)) = (\exists x)(\sim F(x));$ $\quad \sim((\exists x)F(x)) = (\forall x)(\sim F(x)).$

14) $(\forall x)F(x) \wedge (\forall x)H(x) = (\forall x)(F\,(x) \wedge H(x));$

$(\exists x)F(x) \vee (\exists x)H(x) = (\exists x)(F(x) \vee H(x)).$

Переменная в формулах логики предикатов может быть переименована. На использовании этого приема основаны обобщающие законы (п. 13 перечисления) следующие правила преобразования:

15. $(Q_1x)F(x) \vee (Q_2x)H(x) = (Q_1x)F(x) \vee (Q_2z)H(z) = (Q_1x)\,(Q_2z)\,(F\,(x) \vee H(z)),$

$(Q_3x)F(x) \wedge (Q_4x)H(x) = (Q_3x)F(x) \wedge (Q_4z)H(z) = (Q_3x)\,(Q_4z)\,(F\,(x) \wedge H(z)).$

Любая формула логики предикатов допускает эквивалентную предваренную нормальную форму.

Введение кванторов в формулы логики предикатов позволило увеличить выразительную мощность, одновременно усложнив язык данного формализма. Упрощение языка возможно за счет приемлемого уменьшения выразительной мощности. Идея упрощения состоит в том, чтобы, сохранив противоречивость формулы, в ней исключить

кванторы существования путем использования сколемовских функций. Тогда произвольной формуле **F** логики предикатов сопоставляется некоторая формула $\mathbf{S_F}$, причем, если формула **F** невыполнима, то и формула $\mathbf{S_F}$ также невыполнима. Такая связь между формулами **F** и $\mathbf{S_F}$ строго слабее, чем логическая эквивалентность, однако именно такое сопоставление позволяет провести доказательство невыполнимости формулы более эффективно.

*Сколемовской стандартной формой* называется формула, находящаяся в предваренной форме, у которой матрица приведена к конъюнктивной нормальной форме. Получить сколемовскую стандартную форму из произвольной формулы логики предикатов можно, используя следующую процедуру.

Шаг 1. Привести данную формулу к предваренной нормальной форме.

Шаг 2. Привести матрицу к конъюнктивной нормальной форме.

Шаг 3. Если квантора существования в префиксе нет, то полученная формула и есть сколемовская стандартная форма, в противном случае, выбирается самый левый кванторный комплекс существования в префиксе $\mathbf{(\exists x)}$.

Шаг 4. Если никакой квантор всеобщности не стоит в префиксе левее выбранного квантора существования, то, выбрав новую константу **a**, отличную от других констант, входящих в матрицу, заменим все **x** в матрице на константу **a**, вычеркнем кванторный комплекс существования $\mathbf{(\exists x)}$ из префикса. Перейти на шаг 3.

Шаг 5. Если в префиксе левее выбранного кванторного комплекса существования $\mathbf{(\exists x)}$ стоят кванторные комплексы всеобщности $\mathbf{(\forall y_1) \ldots (\forall y_m)}$, т.е. выбранный квантор существования находится в области действия кванторов всеобщности, что означает наличие зависимости **x** от $\mathbf{y_1, \ldots y_m}$, тогда выбрав новый *m*-местный функциональный символ **f**, отличный от других функциональных символов, входящих в матрицу, заменим все **x** в матрице на функцию $\mathbf{f(y_1, \ldots y_m)}$, вычеркнем кванторный комплекс существования $\mathbf{(\exists x)}$ из префикса. Перейти на шаг 3.

Константы и функции, используемые для замены переменных квантора существования, называются *сколемовскими функциями*.

Рассмотрим пример, пусть необходимо получить сколемовскую стандартную форму формулы

$$(\forall x)(\exists y)P(x, y) \to (\forall z)(\exists v)((\forall w)Q(z, w) \to (\exists w)R(z, v, w)) =$$
$$\sim((\forall x)(\exists y)P(x, y)) \vee (\forall z)(\exists v)((\forall w)Q(z, w) \to (\exists w)R(z, v, w)) =$$
$$\sim((\forall x)(\exists y)P(x, y)) \vee (\forall z)(\exists v)(\sim((\forall w)Q(z, w)) \vee (\exists w)R(z, v, w)) =$$
$$((\exists x)(\forall y)\sim P(x, y)) \vee (\forall z)(\exists v)(\sim((\forall w)Q(z, w)) \vee (\exists w)R(z, v, w)) =$$
$$(\exists x)(\forall y)(\sim P(x, y)) \vee (\forall z)(\exists v)((\exists w)(\sim Q(z, w)) \vee (\exists w)R(z, v, w)) =$$

$$(\exists x)(\forall y)(\sim P(x, y)) \vee (\forall z)(\exists v)(\exists w)(\sim Q(z, w) \vee R(z, v, w)) =$$
$$(\exists x)(\forall y)(\forall z)(\exists v)(\exists w)(\sim P(x, y) \vee \sim Q(z, w) \vee R(z, v, w)) =$$
$$\{x = a, v = f(y, z), w = g(y, z)\} =$$
$$(\forall y)(\forall z)(\sim P(a, y) \vee \sim Q(z, g(y, z)) \vee R(z, f(y, z), g(y, z))).$$

Сколемовская стандартная форма — это предваренная форма, префикс которой содержит только кванторы всеобщности. Поэтому удобно префикс опустить, считая, что каждая переменная в матрице управляется квантором всеобщности, а так как конъюнктивная форма – это конъюнкция дизъюнктов, то сколемовскую стандартную форму можно представить в виде множества дизъюнктов.

📖 ***Сколемовская стандартная форма — это множество дизъюнктов.***

Например, стандартная форма

$$(\forall y)(\forall z)(\sim P(a, y) \vee \sim Q(z, g(y, z)) \vee R(z, f(y, z), g(y, z)))$$

может быть представлена множеством

$$S = \{\sim P(a, y) \vee \sim Q(z, g(y, z)) \vee R(z, f(y, z), g(y, z))\}.$$

## *Задание*

I. Ниже приведенные формулы означают следующее:

L(x) – x является юристом, J(x) – x является судьей,
S(x) – x является жуликом, O(x) – x является стариком,
V(x) – x бодрый, P(x) – x является политиком,
C(x) – x является сенатором.

Перевести следующие выражения на русский язык:

1. $(\forall x)(\ J(x) \to L(x))$ 2. $(\exists x)\ (L(x) \wedge S(x))$
3. $(\forall x)(\ J(x) \to \sim S(x))$ 4. $(\exists x)(S(x) \wedge V(x))$
5. $\sim(\forall x)(L(x) \to J(x))$ 6. $(\exists x)(L(x) \wedge P(x) \wedge C(x))$
7. $(\forall x)(\ C(x) \to \sim V(x))$ 8. $(\forall x)(\ (C(x) \wedge O(x)) \to L(x))$

II. Доказать общезначимость следующих формул:

$$(\exists x)\ (\exists y)P(x, y) \leftrightarrow (\exists y)\ (\exists x)\ P(x, y);$$
$$(\exists x)\ (\forall y)P(x, y) \to (\forall y)\ (\exists x)\ P(x, y);$$

III. Для следующей интерпретации
D={**a,b**}

| P(a, a) | P(a, b) | P(b, a) | P(b, b) |
|---|---|---|---|
| И | Л | И | Л |

определить истинностные значения следующих формул

1. $(\exists y)(\sim P(a, y))$ 2. $(\forall x)\ (\forall y)\ P(x, y) \to P(y, x))$

3. **(∀x) (∃y)(~P(x, y))** 4. **(∀x) (∀y) P(x, y)**
5. **(∀x) P(x, x)** 6. **(∃x) (∀y) P(x, y) → ~P(y, x))**

IV. Преобразовать данные формулы в предваренную и сколемовскую нормальную форму.

**(∃x)(∀y)(~P(x, y) → (∀z) (Q(z))**
**(∃x)(~((∀y)(P(x, y)) → ((∃z)Q(z) ↔R(x)))**
**(∀x)(∀y)((∃z)(P(x, y,z) ↔ ((∃u)Q(x,u) → (∃v)Q(y,u)))**

## *Методические указания*

Разберитесь с синтаксисом логики предикатов. Уясните основные синтаксические отличия логики высказываний и логики предикатов.

Раскройте содержание понятия «интерпретация формулы логики предикатов первого порядка», обратите внимание на отличия в правилах интерпретации формул логики предикатов и формул логики высказываний.

## *Самостоятельная работа*

I. Для следующей интерпретации
D={**a,b**}

| **P(a, a)** | **P(a, b)** | **P(b, a)** | **P(b, b)** |
|---|---|---|---|
| **Л** | **И** | **И** | **Л** |

определить истинностные значения следующих формул

1. **(∃y)(~P(a, y))** 2. **(∀x) (∀y) P(x, y) → P(y, x))**
3. **(∀x) (∃y)(~P(x, y))** 4. **(∀x) (∀y) (P(x, y) ∧ P(y, x))**
5. **(∀x) P(x, x)** 6. **(∃x) (∀y) P(x, y) ↔ ~P(y, x))**

II. Для следующей интерпретации
D={**a,b**} **f(a)=b; f(b)= a**;

| **P(a, a)** | **P(a, b)** | **P(b, a)** | **P(b, b)** | **Q(a)** | **Q(b)** |
|---|---|---|---|---|---|
| **И** | **Л** | **И** | **Л** | **И** | **Л** |

определить истинностные значения следующих формул

1. **(∃y)(~P(f(y), f(b)) ↔ Q(y))** 2. **(∀x) (∀y) P(x, y) → Q(x))**
3. **(∀x) (~P(x, y) ∨ Q(x))** 4. **(∀x) (∀y) (P(x, y) ∧ Q(y))**
5. **(∃x) (P(x, x) → Q(x))** 6. **(∃x) (∀y) P(x, y) ↔ ~Q(y))**

III. Преобразовать данные формулы в предваренную и сколемовскую нормальную форму.

**(∀x)(∀y)(~P(x, y) → (∃x)(Q(x))**

**$(\exists x)(\sim((\exists y)(P(x, y)) \rightarrow ((\exists z)Q(z) \rightarrow R(x)))$**
**$(\forall x)(\forall y)((\exists z)(P(x,y,z) \wedge ((\exists u)Q(x,u) \rightarrow (\exists v)Q(y,u)))$**

## *Контрольные вопросы*

1. Что такое правильно построенная формула в логике предикатов?
2. Как задается интерпретация в логике предикатов?
3. Как получить сколемовскую стандартную форму?

# ЗАНЯТИЕ №4. ВЫВОДЫ В ЛОГИЧЕСКИХ МОДЕЛЯХ ПЕРВОГО ПОРЯДКА

*Цель занятия* – освоить приемы и способы логического вывода в логике предикатов.

## *H-интерпретация*

Формула логики предикатов противоречива в том случае, когда противоречиво множество дизъюнктов, представляющих стандартную форму этой формулы, а противоречиво множество дизъюнктов тогда, когда оно ложно при всех интерпретациях во всех предметных областях.

📖 ***Черч и Тьюринг: Не существует общего универсального метода, позволяющего определить общезначимость или противоречивость формулы логики предикатов первого порядка.***

📖 ***Однако существуют алгоритмы, подтверждающие общезначимость или противоречивость формулы, если формула действительно общезначима или противоречива. Для необщезначимых или непротиворечивых формул алгоритмы в общем случае свою работу не заканчивают.***

Частные, но интересные с точки зрения определения процедуры поиска доказательства противоречивости формулы, результаты были получены Эрбраном. Так как рассмотрение всех возможных интерпретаций формулы логики предикатов в общем случае невозможно, Эрбраном была найдена специальная универсальная область интерпретации. Стандартная форма формулы противоречива тогда и только тогда, когда форма ложна при всех интерпретациях в этой области. Называют эту область *эрбрановским универсумом* и определяется она для множества дизъюнктов **S** следующим образом:

- множество констант нулевого уровня $\mathbf{H_0}$ состоит из констант, встречающихся в **S**; если **S** не содержит констант, тогда $\mathbf{H_0}$ содержит одну произвольно выбранную константу, допустим $\mathbf{H_0 = \{c\}}$;
- множество констант **i**-го уровня (i = 1, 2, …, ∞) определяется как объединение констант уровня **i –1** и множества всех термов $\mathbf{f(t_1, t_2, \ldots, t_n)}$, где **f** — все функциональные символы, встречающиеся в **S**, а $\mathbf{t_1, t_2, \ldots, t_n}$ константы уровня **i -1**;
- множество $\mathbf{H_\infty}$ есть эрбрановский универсум множества дизъюнктов **S.**

Элементы эрбрановского универсума — это абстрактные объекты, не имеющие конкретной интерпретации. Если во множестве **S** отсутствуют функции, то эрбрановский универсум всегда конечен и состоит

из множества констант. Если же множество **S** содержит хотя бы одну функцию, то универсум всегда бесконечен.

Рассмотрим несколько примеров. Пусть множество дизъюнктов

**S = {P(x, y) ∨ ~Q(x, z, u), ~P(u, y) ∨ R(y) ∨ Q(x, y, u), S(x)},** тогда $\mathbf{H_\infty = \{c\}}$.

Для множество дизъюнктов

**S** = {**Q(a, g(y)), P(x, y)**}, универсум $\mathbf{H_\infty = \{a, g(a), g(g(a)), g(g(g(a))), \ldots\}}$.

Если множество дизъюнктов

**S = {~P(a, y) ∨ ~Q(z, g(y, z)) ∨ R(z, f(y, z), g(y, z))}, то**

$\mathbf{H_\infty}$ **={a, g(a, a), f(a, a), g(a, g(a, a)), g(g(a, a), a), g(g(a, a), g(a, a)), g(a, f(a, a)), g(f(a, a), a), g(f(a, a), f(a, a)), f(a, g(a, a)), f(g(a, a), a), f(g(a, a), g(a, a)), …}.**

*Эрбрановской интерпретацией* или *H-интерпретацией* для множества дизъюнктов **S** называется интерпретация, удовлетворяющая следующим условиям:

- предметной областью является эрбрановский универсум;
- интерпретация отображает все константы из **S** в соответствующую эрбрановскую константу;
- если **f** — **n**-местный функциональный символ и $\mathbf{h_1, h_2, \ldots, h_n}$ — константы эрбрановского универсума, то в эрбрановской интерпретации через **f** обозначается функция, отображающая $\mathbf{(h_1, h_2, \ldots, h_n)}$ в $\mathbf{f(h_1, h_2, \ldots, h_n)}$.

В общем случае эрбрановских интерпретаций на множестве **S** может быть бесконечно много, так как интерпретации предикатов и функций могут быть выбраны произвольно. Пусть множество дизъюнктов

**S** = {**P(x, y) ∨ ~Q(x, z, u)**, **~P(u, y) ∨ R(y) ∨ Q(x, y, u), S(x)**}, эрбрановский универсум для этого множества дизъюнктов $\mathbf{H_\infty = \{c\}}$. Некоторые интерпретации приведены ниже:

$\mathbf{I_1}$ = {**P(c, c), Q(c, c, c), R(c), S(c)**}, $\mathbf{I_2}$ = {**~P(c, c), Q(c, c, c), R(c), S(c)**},
$\mathbf{I_3}$ = {**P(c, c), ~Q(c, c, c), R(c), S(c)**}, $\mathbf{I_4}$ = {**P(c, c), Q(c, c, c), ~R(c), S(c)**}.

Так как здесь имеется четыре атома **P(c, c), Q(c, c, c), R(c), S(c)**, то всего существует $2^4 = 16$ эрбрановских интерпретаций.

Для множества дизъюнктов **S** = {**Q(a, g(y)), P(x, y)**} эрбрановский универсум бесконечен $\mathbf{H_\infty = \{a, g(a), g(g(a)), g(g(g(a))), \ldots\}}$, и, соответственно, эрбрановских интерпретаций будет бесконечное множество, четыре из них приведены ниже:

$\mathbf{I_1}$ **= {P(a, a), Q(a, a), P(a, g(a)), Q(a, g(a)), P(g(a), a), Q(g(a), a), P(g(a), g(a)), Q(g(a), g(a)), …},**

$\mathbf{I_2}$ **= {~P(a, a), Q(a, a), ~P(a, g(a)), Q(a, g(a)), ~P(g(a), a), Q(g(a), a), ~P(g(a), g(a)), …},**

**$I_3$ = {P(a, a), ~Q(a, a), P(a, g(a)), ~Q(a, g(a)), P(g(a), a), ~Q(g(a), a), P(g(a), g(a)), …},**

**$I_4$ = {~P(a, a), ~Q(a, a), ~P(a, g(a)), ~Q(a, g(a)), ~P(g(a), a), ~Q(g(a), a), ~P(g(a), g(a)), …}.**

Доказано, что множество дизъюнктов **S** невыполнимо тогда и только тогда, когда оно ложно при всех своих эрбрановских интерпретациях.

📖 ***Для проверки выполнимости множества дизъюнктов необходимо рассматривать только эрбрановские интерпретации***.

*Выражение* — это терм, множество термов, множество атомов, множество дизъюнктов. Если выражение не содержит переменных, то оно называется *основным*.

*Основной пример* дизъюнкта **C** множества дизъюнктов **S** есть дизъюнкт, полученный заменой переменных в **C** на константы эрбрановского универсума **S**. Пусть множество дизъюнктов **S** = {**Q(a, g(y)), P(x, y)**}, дизъюнкт **C** = **P(x, y)**, эрбрановский универсум $\mathbf{H_\infty}$ = {**a, g(a), g(g(a)), g(g(g(a))), …**}, тогда основной пример **C' = P(g(a), a)**.

*Основной пример* выполняется в данной интерпретации тогда и только тогда, когда в этом основном примере существует основная литера, которая есть и в данной интерпретации. Пусть множество дизъюнктов **S** = {**Q(a, g(y)), P(x, y)**}, дизъюнкт **C** = **P(x, y)**, основной пример **C' = P(g(a), a)**, шесть интерпретаций:

**$I_1$ = {P(a, a), Q(a, a), P(a, g(a)), Q(a, g(a)), P(g(a), a), Q(g(a), a), P(g(a), g(a)), Q(g(a), g(a)), …}**,

**$I_2$ = {~P(a, a), Q(a, a), ~P(a, g(a)), Q(a, g(a)), ~P(g(a), a), Q(g(a), a), ~P(g(a), g(a)), …}**,

**$I_3$ = {P(a, a), ~Q(a, a), P(a, g(a)), ~Q(a, g(a)), P(g(a), a), ~Q(g(a), a), P(g(a), g(a)), …}**,

**$I_4$ = {~P(a, a), ~Q(a, a), ~P(a, g(a)), ~Q(a, g(a)), ~P(g(a), a), ~Q(g(a), a), ~P(g(a), g(a)), …}**,

**$I_5$ = {~P(a, a), Q(a, a), P(a, g(a)), Q(a, g(a)), P(g(a), a), Q(g(a), a), P(g(a), g(a)), …},**

**$I_6$ = {~P(a, a), ~Q(a, a), P(a, g(a)), Q(a, g(a)), P(g(a), a), Q(g(a), a), P(g(a), g(a)), …}**,

тогда основной пример **C'** выполняется в интерпретации **$I_1$, $I_3$, $I_5$, $I_6$** и опровергается в **$I_2$, $I_4$**.

*Дизъюнкт выполняется* в данной интерпретации тогда и только тогда, когда каждый основной пример выполняется в данной интерпретации.

*Дизъюнкт опровергается* в данной интерпретации тогда и только тогда, когда существует хотя бы один основной пример данного дизъюнкта, который не выполняется в данной интерпретации.

Вернемся к предыдущему примеру, дизъюнкт **C = P(x, y)** выполняется в интерпретации $\mathbf{I_1}$, $\mathbf{I_3}$ и опровергается в $\mathbf{I_2}$, $\mathbf{I_4}$, $\mathbf{I_5}$, $\mathbf{I_6}$.

*Множество дизъюнктов невыполнимо* тогда и только тогда, когда для каждой интерпретации существует хотя бы один основной пример дизъюнкта, невыполнимый в данной интерпретации.

Пусть **S** = {**P(x)** ∨ **~Q(x)**, ~**P(x)**, **Q(x)**}, на данном множестве дизъюнктов существуют четыре интерпретации:

$\mathbf{I_1}$ = {**P(c), Q(c)**},
$\mathbf{I_2}$ = {~**P(c), Q(c)**},
$\mathbf{I_3}$ = {**P(c), ~Q(c)**},
$\mathbf{I_4}$ = {~**P(c), ~Q(c)**}.

В интерпретации $\mathbf{I_1}$ опровергается дизъюнкт ~**P(x)**; дизъюнкт **P(x)** ∨ **~Q(x)** опровергается в интерпретации $\mathbf{I_2}$; интерпретация $\mathbf{I_3}$ опровергает дизъюнкт **Q(x)**; этот же дизъюнкт **Q(x)** опровергается в интерпретации $\mathbf{I_4}$. Рассматриваемое множество дизъюнктов невыполнимо, потому что оно опровергается во всех интерпретациях.

## *Теорема Эрбрана и метод резолюций*

📖 ***Теорема Эрбрана. Множество дизъюнктов невыполнимо тогда и только тогда, когда существует конечное невыполнимое множество основных примеров дизъюнктов указанного множества.***

Рассмотрим предыдущий пример, **S** = {**P(x)** ∨ **~Q(x)**, ~**P(x)**, **Q(x)**}, приведенное множество дизъюнктов невыполнимо потому, что существует следующее невыполнимое множество основных примеров:

**S'** = {**P(c)** ∨ **~Q(c)**, ~**P(c)**, **Q(c)**}.

На основе теоремы Эрбрана может быть создана компьютерная процедура, генерирующая множества основных примеров и проверяющая их невыполнимость. Одним из первых, кто осуществил это, был П. Гилмор. Он в 1960 году написал компьютерную программу, которая порождала множества основных примеров, полученных заменой переменных во множестве дизъюнктов на константы эрбрановского универсума. Так как множество основных примеров — это конъюнкция дизъюнктов, не содержащих переменных (по сути, это высказывания), то можно воспользоваться любым из восьми способов проверки противоречивости данного множества при условии, что множество основных примеров конечно.

Процедуры поиска опровержения, основанные на теореме Эрбрана, имеют один существенный недостаток — они требуют генерации множеств $\mathbf{S'_1, S'_2, \ldots, S'_n, \ldots}$ основных примеров рассматриваемого множества дизъюнктов **S**. Очень часто размерность множеств основных примеров растет экспоненциально, по этой причине такие процедуры не имеют практического применения на современных компьютерах. И только с появлением метода резолюций, разработанного Дж. Робинсоном, были найдены эффективные компьютерные процедуры доказательства невыполнимости множества дизъюнктов. Идея метода Робинсона состоит в том, чтобы работать напрямую с дизъюнктами, входящими в рассматриваемое множество, не порождая основных примеров. Реализация метода резолюций требует применения операции унификации, но прежде чем объяснить эту операцию, дадим определение понятия подстановки.

*Подстановка* — это конечное множество вида $\{v_1 = t_1, v_2 = t_2, \ldots, v_n = t_n\}$, где $v_j$ — это переменная, а $t_j$ — это терм, отличный от $v_j$, причем все $v_j$ отличны друг от друга.

*Унифицировать* два или более выражений значит найти такую подстановку, которая делает выражения одинаковыми (тождественными). Такая подстановка называется *унификатором*.

Рассмотрим два выражения **P(x, a, f(a, g(y)))** и **P(h(v), z, f(z, u))**. Они не тождественны, однако, если применить к ним операцию унификации с подстановкой $\{x = h(v), z = a, u = g(y)\}$, то получим два тождественных выражения **P(h(v), a, f(a, g(y)))**.

В логическом исчислении удобно работать с дизъюнктами, не содержащими повторяющихся литер. Устранение повторения в результирующем дизъюнкте выполняют с помощью операции склейки. Если две или более литер с одинаковым знаком в дизъюнкте **C** имеют общий унификатор $\{v_1 = t_1, v_2 = t_2, \ldots, v_n = t_n\}$, то дизъюнкт, полученный из **C** заменой одновременно всех вхождений переменных $v_i$ на термы $t_i$, называется *склейкой* дизъюнкта **C**.

Применение операции унификации и склейки позволяет использовать метод резолюций в логике предикатов первого порядка.

Пусть $\mathbf{C_1}$ и $\mathbf{C_2}$ — дизъюнкты, не имеющие общих переменных. Если в дизъюнкте $\mathbf{C_1}$ существует литера $\mathbf{L_1}$, а в дизъюнкте $\mathbf{C_2}$ существует литера $\mathbf{\sim L_2}$, и литеры $\mathbf{L_1}$ и $\mathbf{\sim L_2}$ унифицируемы, то, вычеркнув литеры $\mathbf{L_1}$ и $\mathbf{\sim L_2}$ из $\mathbf{C_1}$ и $\mathbf{C_2}$ соответственно построим дизъюнкцию оставшихся дизъюнктов, которая называется *бинарной резольвентой* дизъюнктов $\mathbf{C_1}$ и $\mathbf{C_2}$.

Рассмотрим пример, пусть $\mathbf{C_1 = P(x, g(b)) \vee \sim Q(x)}$ и $\mathbf{C_2 = \sim P(a, g(x)) \vee R(z)}$. Сначала переименуем переменную **x** в первом дизъюнкте:

$\mathbf{C_1 = P(y, g(b)) \vee \sim Q(y)}$. Выберем в дизъюнкте $\mathbf{C_1}$ литеру $\mathbf{L_1=P(y, g(b))}$ и литеру $\mathbf{L_2= \sim P(a, g(x))}$ в $\mathbf{C_2}$. Литеры $\mathbf{L_1}$ и $\sim\mathbf{L_2}$ имеют общий унификатор {y = a, x = b}, тогда бинарная резольвента имеет следующий вид: $\mathbf{\sim Q(a)) \vee R(z)}$.

Резольвентой дизъюнктов $\mathbf{C_1}$ и $\mathbf{C_2}$ является одна из следующих резольвент:

бинарная резольвента $\mathbf{C_1}$ и $\mathbf{C_2}$;

бинарная резольвента $\mathbf{C_1}$ и склейки $\mathbf{C_2}$;

бинарная резольвента $\mathbf{C_2}$ и склейки $\mathbf{C_1}$;

бинарная резольвента склейки $\mathbf{C_1}$ и склейки $\mathbf{C_2}$.

Пусть $\mathbf{C_1 = \sim P(x, y) \vee \sim P(f(z), z) \vee \sim Q(x, y)}$ и $\mathbf{C_2 = P(f(g(a)), g(a)) \vee R(a)}$. Склейка $\mathbf{C_1}$ имеет следующий вид: $\mathbf{\sim P(f(z), z) \vee \sim Q(f(z), z).}$ Бинарная резольвента склейки $\mathbf{C_1}$ и $\mathbf{C_2}$ равна $\mathbf{\sim Q(f(g(a)), g(a)) \vee R(a)}$.

## *Задание*

I. Пусть $\mathbf{S = \{P(f(x), a, g(f(x),b))\}}$.

1) Найти $\mathbf{H_0}$ и $\mathbf{H_1}$.

2) Найти все основные примеры на $\mathbf{H_0}$ и $\mathbf{H_1}$.

II. Пусть $\mathbf{S = \{P(x), \sim P(f(y))\}}$.

1) Найти $\mathbf{H_0}$, $\mathbf{H_1}$, $\mathbf{H_2}$ и $\mathbf{H_3}$.

2) Возможно ли найти интерпретацию, которая опровергает **S**? Если да, то приведите одну. Если нет, то почему?

III. Пусть $\mathbf{S = \{P(x, a, g(x, b)), \sim P(f(y), z, g(f(a),b))\}}$.

Найти невыполнимое множество **S'** основных примеров дизъюнктов в **S**.

IV. Ни один человек не является четвероногим. Все женщины – люди. Следовательно, ни одна женщина не является четвероногой. Доказать.

V. Ни один первокурсник не любит второкурсников. Все, живущие на шестом этаже, – второкурсники. Следовательно, ни один первокурсник не любит никого из живущих на шестом этаже. Доказать.

VI. Ни один торговец наркотиками не является наркоманом. Некоторые наркоманы привлекались к ответственности. Следовательно, некоторые люди, привлекавшиеся к ответственности, не являются торговцами наркотиками. Доказать.

VII. Некоторые пациенты любят своих докторов. Ни один пациент не любит знахаря. Следовательно, никакой доктор не является знахарем. Доказать.

### *Методические указания*

Изучите способ получения универсальной области интерпретации формул логики предикатов. Разберитесь с тем, как может быть использована теорема Эрбрана в логическом выводе.

Уясните принципиальные отличия в использовании теоремы Эрбрана и метода резолюций в логическом выводе.

### *Самостоятельная работа*

I. Пусть **S = {P(f(x), a, g(f(b)))}**.
   1) Найти $\mathbf{H_0}$ и $\mathbf{H_1}$.
   2) Найти все основные примеры на $\mathbf{H_0}$ и $\mathbf{H_1}$.

II. Пусть **S = {P(x), ~P(f(y))∨Q(z), ~Q(x)}**.
   1) Найти $\mathbf{H_0}$, $\mathbf{H_1}$, $\mathbf{H_2}$ и $\mathbf{H_3}$.
   2) Возможно ли найти интерпретацию, которая опровергает **S**? Если да, то приведите одну. Если нет, то почему?

III. Пусть **S = {P(x), Q(x,f(x)) ~P(x), Q(g(y),z)}**.
   Найти невыполнимое множество **S'** основных примеров дизъюнктов в **S**.

IV. Все первокурсники встречаются со всеми второкурсниками. Ни один первокурсник не встречается ни с одним студентом предпоследнего курса. Существуют первокурсники. Следовательно, ни один второкурсник не является студентом предпоследнего курса. Доказать.

V. Никакой торговец подержанными автомобилями не покупает подержанный автомобиль для своей семьи. Некоторые люди, покупающие подержанные автомобили для своих семей – жулики. Следовательно, некоторые жулики не являются торговцами подержанными автомобилями. Доказать.

### *Контрольные вопросы*

1. Что есть эрбрановский универсум множества дизъюнктов?
2. Могут ли быть использованы способы логического вывода, определенные в логике высказываний, для вывода в логике предикатов? Если «да», то опишите, как это можно сделать; если «нет», то объясните, почему невозможно.
3. Предложите компьютерный алгоритм применения теоремы Эрбрана и метода резолюций для доказательства невыполнимости множества дизъюнктов.

# ЗАНЯТИЕ №5. ЗНАКОМСТВО С VISUAL PROLOG. ОБРАБОТКА СПИСКОВ.

*Цель занятия* – знакомство со средой Visual Prolog; изучение операций над списками, построение деревьев вывода.

## *Знакомство с языком Visual Prolog*

*Основные элементы языка Visual Prolog.*

Visual Prolog – логический и объектно-ориентированный язык программирования, основой которого являются классический язык ПРОЛОГ, а также языки Turbo Prolog и PDC Prolog. В ходе выполнения практических работ основное внимание будет уделено только логическим аспектам языка.

Основные конструкции языка Visual Prolog – термы и утверждения. Утверждение – это факт, правило, вопрос. Терм − объект данных. Термом является константа, переменная, сложный терм. Для обозначения термов используются имена. Имя может начинаться с любой латинской буквы или символа "_", затем следует любая комбинация букв, цифр и "_". Необходимо помнить, что имена символьных констант должны начинаться с маленькой буквы; имена переменных должны начинаться с большой буквы или символа "_". Кроме того, в системе существуют зарезервированные имена - это наименования программных секций, имена стандартных предикатов и операторов, команды компилятора. Для задания анонимной переменной используется символ "_". Такая переменная используется в том случае, когда значение этой переменной не имеет значения.

Сложный терм состоит из имени и аргументов. Список – это специальный вид сложного терма, состоящий из головы списка и хвоста списка, где голова - это элемент списка, а хвост – это список. Синтаксически список задается следующим образом: [X|L], где X – голова, L – хвост.

*Простые типы данных.*

Имеется шесть следующих простых типов данных:

**Char** – знаковый тип. Отдельный символ, заключенный в апострофы.

**Integer** – целое число в диапазоне от -32768 до 32767.

**Real** – действительный тип.

**String** – строковый тип. Последовательность символов, заключенная в кавычки.

**Symbol** – символьный тип. Последовательность букв, цифр и знаков подчеркивания, которая начинается со строчной буквы или заключена в кавычки.

**File** – файловый тип. Имя файла.

Данные типа **symbol** в отличие от данных типа **string** запоминаются в таблице символов. Таблица символов размещается в оперативной памяти, поэтому ее использование обеспечивает наиболее быстрый поиск.

*Программные секции.*

Программа на Visual Prolog может состоять из следующих секций:

**domains**
**global domains**
**dataBase**
**predicates**
**global predicates**
**goal**
**clauses**

Префикс **global** с именами **domains** и **predicates** используется в модульном программировании и означает, что нижеследующие объявления относятся к нескольким программам.

Программа может содержать несколько разделов **DOMAINS, PREDICATES** и **CLAUSES**. При этом необходимо соблюдать следующие ограничения:

- в программе может использоваться только один раз **goal**;
- все предикаты в **CLAUSES** с одинаковыми именами должны записываться подряд;
- раздел **GLOBAL PREDICATES** в программе может встречаться только один раз, причем впереди данного раздела не должны стоять разделы **PREDICATES**;
- разделы, содержащие предикаты базы данных **DATABASE**, должны предшествовать объявлениям **GLOBAL PREDICATES** и **PREDICATES**.

Ниже остановимся на описании отдельных секций.

**DOMAINS** объявляется одним из следующих способов:

1. name1[,name2[,...nameN]] = declaration, где declaration – один из простых типов;
2. listdom = name*, где listdom – список элементов, name – это элемент, либо ранее описанный в **DOMAINS**, либо один из простых типов.
3. term = f1(t11,...,t1N); - описание сложного составного терма;

4. file = name1; name2,...; nameN, данный формат используется для задания файлу символического имени.

В разделе **PREDICATES** указывается имя предиката и область его аргументов или простых типов:

{procedure | determ | nondeterm | multi} pred1(.....)
{procedure | determ | nondeterm | multi} pred2(.....)
**.....**
{procedure | determ | nondeterm | multi} predN(.....)

Здесь допускается многократное объявление предиката с одним и тем же именем, в зависимости от типов обрабатываемых данных и числа аргументов.

В определении предикатов могут использоваться ключевые слова для описания режимов поведения предикатов:

*multi* определяет недетерминированные предикаты, которые могут генерировать множественные решения. Предикаты, объявленные с данным ключевым словом никогда не терпят неудачу и, поэтому всегда генерируют по крайней мере одно решение.

*nondeterm* определяет недетерминированное поведение предикатов, которые могут генерировать множественные решения. Предикаты, объявленные с ключевым словом nondeterm могут терпеть неудачу.

*procedure* определяет предикаты называемые процедурами. Процедуры никогда не терпят неудачу и всегда имеют только одно решение.

*determ* определяет детерминированные предикаты, эти предикаты имеют только одно решение.

Раздел **CLAUSES** содержит набор правил и фактов, где правило имеет следующую структуру:

pred1(.....) :– p1(....), p2(....), **...** , pN(....).

Факт имеет вид: pred1(....).

Причем все встречающиеся в разделе **CLAUSES** предикаты должны быть описаны в разделе **PREDICATES** либо должны быть стандартными.

*Поточные шаблоны.*

Большинство стандартных предикатов выполняют несколько функций в зависимости от состояния параметров, входящих в предикат. В данном случае заранее определен какой-либо один параметр, в другом случае известны другие параметры, иногда к моменту обращения могут быть известны все параметры. Известные параметры предиката называются входными, неизвестные - выходными. Совокупность входных параметров определяет работу предиката. Эта совокуп-

ность называется поточным шаблоном. Если предикат имеет два аргумента, то вызываться он может с четырьмя вариантами поточного шаблона:

**(i,i) (i,o) (o,i) (o,o)**

где **i** - входной параметр, **o** - выходной параметр.

Однако не для каждого предиката все возможные варианты поточного шаблона имеют смысл.

Пример использования поточного шаблона для описания предиката *append* приведен ниже.

```
domains
  li=integer*
predicates
  procedure append(li,li,li) - (i,i,o)
  determ      (i,i,i)
  nondeterm (o,o,i)
```

## *Знакомство со средой Visual Prolog*

Язык Visual Prolog встроен в интегрированную среду разработки. При запуске системы открывается окно (рис.1.1), в верхней части которого имеется меню и панель инструментов. Нижняя часть окна содержит строку, служащую для вывода кратких подсказок по использованию меню, окон, панели инструментов.

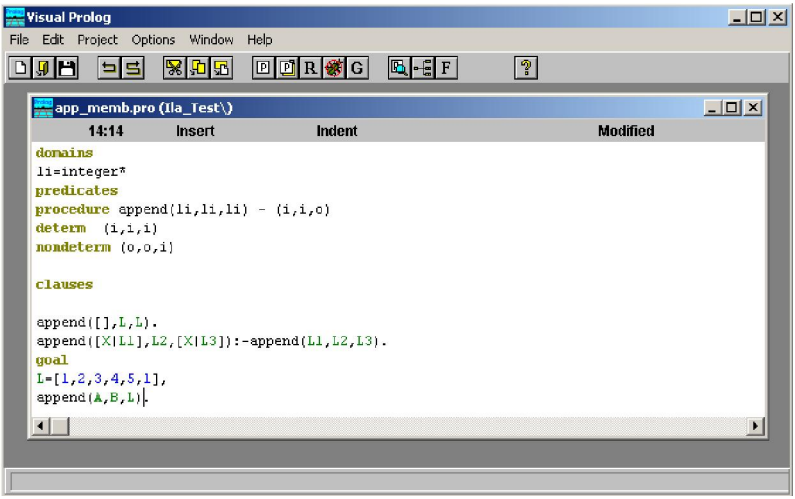



Рис.1. Главное окно среды Visual Prolog

Выбор той или иной операции меню выполняется стандартным для приложений MS Windows образом. Панель инструментов содержит кнопки, позволяющие осуществить доступ к наиболее часто используемым командам и операциям.

Среда Visual Prolog имеет встроенный редактор исходных текстов программ, основанный на многооконном пользовательском интерфейсе. Для создания файла с исходным текстом программы необходимо выбрать пункт меню «файл» (File) и в нём подпункт «новый» (New). В результате будет создан файл с именем по умолчанию noname* , где * это номер обеспечивающий уникальность имени создаваемого файла. После того как файл был создан, необходимо сохранить его под каким-либо реальным именем на диске, для этого можно в меню «файл» выбрать подпункт «сохранить как» (save as), после чего выведется стандартно окно для сохранения файла.

Для выполнения программы необходимо выбрать пункт меню «Project» и в нём подпункт «Test goal», либо нажать комбинацию клавиш «Ctrl+G», либо нажать кнопку «G» на панели инструментов. После этого откроется окно, в котором выведется результат.

## *Операции над списками*

1) *Операция определение вхождения отдельного элемента в список*:

**member(X, [X | _]).**
**member(X, [_ | Y]):- member(X, Y).**

Ответом на вопрос

**?- member(a, [b, a, c]).**

будет «да».

Рассмотрим процесс получения данного ответа. Начальная резольвента ***member(a, [b, a, c]).*** В программе выбирается первое предложение и делается попытка унифицировать его с резольвентой. Попытка заканчивается неудачей, так как переменной ***X*** одновременно не может быть присвоено значение ***a*** (первый аргумент), ***b*** (голова списка второго аргумента). Далее в программе выбирается второе предложение и делается попытка унифицировать его с резольвентой. Попытка удачна, новая и единственная резольвента - ***member(a, [a,c]).*** В программе выбирается первое предложение, и оно успешно унифицируется с резольвентой. Первое предложение программы не имеет тела, резольвента пуста, решение найдено. Но резольвента ***member(a, [a,c])*** может быть унифицирована и со вторым предложением программы, после унификации получаем новую резольвенту ***member(a, [c]).*** Данная резольвента не может быть унифицирована с первым предложени-

ем программы, а вот со вторым предложением унификация заканчивается успешно, новая резольвента - ***member(a,*** **[]** ***),*** которая не может быть унифицирована ни с первым, ни со вторым предложением. Таким образом, был получен один положительный ответ. Графическое представление поиска решения приведено на рис. 1.2.

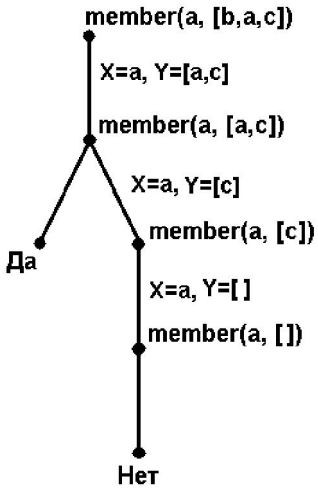



Рис.2. Графическое представление поиска решения

2) *Операция соединения двух списков* для получения третьего (разбиения списка на возможные составляющие):

**append([], L, L).**
**append([X | XL], YL, [X | ZL]):- append(XL, YL, ZL).**

3) *Удаление указанного элемента из списка*:

**delete (_, [], []).**
**delete (X, [X | XL], YL):- delete (X, XL, YL).**
**delete (X, [Z | XL], [Z | YL]):- X<>Z, delete (X, XL, YL).**

4) *Нахождения последнего элемента списка*:

а) **last (X, [X]).**
**last (X,[_ | XL]):- last (X,XL).**

б) **last (X,XL):-append(_, [X], XL).**

5) *Добавление элемента в список в произвольное место*:

**insert(H, L, [H | L]).**
**insert(H, [X | L],[X | V]):-insert(H, L, V).**

6) *Реверс или обращение списка*:

a) **reverse([], []).**
**reverse(L, [B | RL]):- append(A, [B], L), reverse(A, RL).**

b) **reverse(L, R):- reverse1(L, [], R).**
**reverse1([], R, R).**
**reverse1([X | L], S, R):- reverse1(L, [X | S], R).**

## *Задание*

1. Опишите операции над списками в нотации Visual Prolog.
2. Исследуйте на примерах работу каждой операции, постройте деревья вывода для каждой операции (пример построения дерева вывода приведен на рис. 2).
3. Сравните эффективность различных реализаций предикатов: **reverse, last**.

# ЗАНЯТИЕ №6. ПРОГРАММИРОВАНИЕ БАЗ ЗНАНИЙ В СРЕДЕ VISUAL PROLOG

*Цель работы* – создание программной системы, реализующей базу знаний в среде Visual Prolog.

## *Выбор предметной области*

Необходимо выбрать ту предметную область, в которой вы можете выступить в качестве эксперта. Выбор предметной области предполагает четкое (письменное) формулирование цели создания системы.

Примерный перечень предметных областей:

- Расписание занятий.
- Расписание движения транспорта (авто, авиа, железнодорожного).
- Расписание приема врачами в поликлинике.
- Библиотека, фонотека, видеотека.
- Отдел кадров.
- Магазин, склад, хранилище.

## *Формальное описание предметной области*

В качестве формализма для описания должны быть выбраны таблицы. Количество таблиц и число столбцов в таблице неограниченно. Фрагмент таблицы описания предметной области "Расписание занятий" приведен ниже.

| **Ф И О лектора** | **предмет** | **время** | | | **место** | |
|---|---|---|---|---|---|---|
| | | **день недели** | **начало** | **конец** | **корпус** | **аудитория** |
| Ходашинский Илья Александрович | СИИ | среда | 9 | 11 | ФЭТ | 418 |
| … | … | … | .. | … | … | … |

ТРЕБОВАНИЕ К БАЗЕ ЗНАНИЙ:

- произведение количества полей на количество записей должно быть не менее 100 для табличного представления;
- одно и более полей в таблице должно быть списочного типа.

Примеры полей списочного типа: дни недели отправления транспорта (приема врачом в поликлинике); авторы произведения; перечень специальностей, которыми владеет специалист.

### *Представление формального описания предметной области на языке Visual Prolog*

Каждая строка таблицы должна быть описана в виде факта на языке Visual Prolog. Например, в системе "Расписание занятий" должны быть представлены отношения между четырьмя элементами: названием курса, временем, именем лектора и местом проведения занятий. Факт из таблицы на языке Visual Prolog может быть представлен следующим образом:

```
represent("СИИ", time("среда",9,11),
          lector("Ходашинский", "Илья", "Александрович"),
          place("ФЭТ",418)).
```

### *Формулировка запросов и оформление интерфейса*

Система "Расписание занятий" должна выдавать сведения, например, увязывающие читаемый курс и фамилию лектора, день недели и фамилию лектора, занятость аудиторий по времени.

Отношение, увязывающее читаемый курс и фамилию лектора имеет следующий вид:

```
rel1(F,K):-represent(K,_,lector(F,_,_),_).
```

Отношение, увязывающее день недели и фамилию лектора имеет следующий вид:

```
rel2(F,D):-represent(_, time(D,_,_),lector(F,_,_),_).
```

Отношение, увязывающее занятость аудитории во времени имеет следующий вид:

```
rel3(A, K,D,H,K):-  represent(_,time (D,H,K),_,
                    place(K, A)).
```

Программирование повторяющихся операций поиска в базе может быть выполнен при помощи следующей процедуры:

```
proc_repeat:- «повторяемые операции», fail.
proc_repeat.
```

В первом предложении данной процедуры конструкция ***«повторяемые операции»*** содержит несколько предикатов, как определенных в программе, так и системных. Встроенный предикат ***fail,*** после успешного выполнения конструкции ***«повторяемые операции»,*** вызывает возврат назад и реализует повторное выполнение указанной конструкции. В случае неудачного выполнения конструкции ***«повторяемые операции»,*** выполняется второе предложение процедуры, всегда приводящее к успеху.

Используя механизм автоматического перебора, можно собрать данные, удовлетворяющие некоторому свойству, из базы в список для последующей их обработки. Предикатом, выполняющим такие действия, является предикат ***findall(V, P, L)***, который собирает в список ***L*** все объекты ***V***, входящие в предикат ***P***. Например, лектор может читать несколько курсов, тогда задача определения и вывода курсов, читаемых Ходашинским, может быть решена следующим образом:

```
?- findall (X, represent(X, _, lector("Ходашинский", _, _), _), L),
        write(L), nl.
```

## *Задание*

1. Определите предметную область.
2. Дайте краткое описание выбранной предметной области.
3. Сформулируйте цель создания системы.
4. Сделайте формальное описание предметной области.
5. Создайте не менее 5 запросов к системе, в том числе запрос, обрабатывающий списочное поле.

# ЗАНЯТИЕ №7. СОРТИРОВКА

*Цель работы* – реализация методов сортировки списков на языке ПРОЛОГ.

## *Методы сортировки*

Под сортировкой списка подразумевается переупорядочение его элементов в такую последовательность, когда все элементы списка расположены в порядке невозрастания или неубывания.

а) *Сортировка со вставками*. Здесь неупорядоченный непустой список разбивается на голову и хвост, упорядочивается хвост, и далее голова вставляется в упорядоченный хвост так, чтобы получившийся список остался упорядоченным.

```
insert_sort([H|L], S):- insert_sort(L,S1),
                         insert(H, S1, S).
insert_sort([],[]).

insert(H,[A | S1],[A | S2]):- A<H, !, insert(H, S1,S2).
insert(H, S,[H | S]).
```

б) *Сортировка, основанная на методе пузырька*. Суть метода заключается в перестановке местами двух элементов, находящихся в списке в неупорядоченном состоянии, и повторении этого процесса до тех пор, пока перестановки станут ненужными.

```
bubble_sort(L, S):- append(L1,[A1,A2|L2], L),
                    A2<A1, append(L1,[A2,A1|L2], L3),
                    bubble_sort(L3,S), !.
bubble_sort(S,S).
```

в) *Быстрая сортировка*. Программа быстрой сортировки реализует следующий механизм упорядочения: исходный список разбивается на голову и хвост; хвост разбивается на два списка, в первый список помещаются элементы меньшие, чем голова, во второй - остальные; оба списка сортируются; затем упорядоченные списки сливаются вместе.

```
quick_sort([H | L], S):-  split(H, L, L1, L2),
                          quick_sort(L1, S1),
```

```
                                quick_sort(L2, S2),
                                append(S1, [H | S2], S).
quick_sort([], []).

split(H, [A | LS], [A | L1], L2):- A<H, split(H, LS, L1, L2),!.
split(H, [A | LS], L1, [A | L2]):- A>=H, split(H, LS, L1, L2),!.
split(_, [], [], []).
```

### *Задание*

1. Опишите операции сортировки списков в нотации Visual Prolog.
2. Самостоятельно задайте списки разной длины (пять, девять, одиннадцать элементов) и различной степени упорядоченности исходного списка.
3. Исследуйте методы сортировки для списков разной длины и степени упорядоченности исходного списка. Сравните эти методы по эффективности, построив деревья вывода для каждой операции.

# ЗАНЯТИЕ №8. ПРЕДСТАВЛЕНИЕ ГРАФОВ И ПОИСК ПУТИ НА ГРАФЕ

*Цель работы* – изучение способов представления графов средствами ПРОЛОГ и способов определения пути на графе.

## *Представления графов и поиск пути*

*Графом* называется некоторое множество точек (вершин, узлов) и некоторое множество линий (ребер, дуг), соединяющих определенные пары вершин, или, другими словами, граф – это пара

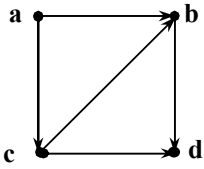
$$G = (N, A),$$

где $N$ – множество вершин и $A$ – множество ребер. Две вершины $p$ и $q$ называются *смежными*, если существует ребро $e = (p, q)$, соединяющее их; а ребро *е инцидентно* $p$ и $q$. *Степенью* вершины $p$ графа $G$ называется число ребер, инцидентных $p$. Если ребра представлены в виде упорядоченных пар вершин $(p, q)$, то такой граф называется *ориентированным*; вершина $p$ – называется началом, $q$ – концом ребра. Если ребра – это неупорядоченные пары вершин, то граф называется *неориентированным*, здесь $(p, q) = (q, p)$. *Маршрутом* в графе называется последовательность вершин $p_1, p_2, p_3, \ldots, p_{N-1}, p_N$, $N \geq 1$, такая, что $(p_i, p_{i+1}) \in A$ при i = 1,2,…,N-1. Маршрут без повторяющихся вершин называется *цепью* или *путем*. Есть другое определение пути, *путем* в графе называется последовательность ребер, в которой конечный узел каждого ребра является в то же время начальным узлом следующего ребра, таким образом, путь из $p_1$ в $p_N$ имеет вид $(p_1, p_2), (p_2, p_3), \ldots, (p_{N-1}, p_N)$. Длина данного пути равна N-1. Граф называется связным, если имеется путь между каждыми двумя вершинами этого графа. Цикл – это путь длины больше 1, который начинается и кончается в одной и той же вершине.

Известно несколько способов представления графа; традиционный – *матрица смежностей*, когда задается квадратная матрица размером $\|N\| \times \|N\|$, в которой элемент $[i, j]$ равен 1, если есть ребро из узла $i$ в узел $j$, и равен 0, в противном случае. Представлять граф в ПРОЛОГе таким способом очень неэффективно. Во-первых, ПРОЛОГ не имеет такой структуры данных, как массив с его быстрым доступом по индексам; во-вторых, применение матрицы смежностей требует памяти объема $\|N\|^2$ даже тогда, когда граф содержит порядка $N$ ре-

бер, и, в-третьих, начальное заполнение матрицы смежностей требует времени порядка $O(\|N\|^2)$.

Представить граф можно с помощью *списков смежностей*, когда для каждого узла графа создается список смежных с ним узлов. Например, граф




можно представить следующим образом:

G = [[a, b, c], [b, d], [c, d, b], [d]].

Здесь принято следующее соглашение: первый элемент подсписка означает узел, остальные элементы подсписка – смежные с ним узлы. Такое представление требует памяти порядка $\|N\| + \|A\|$ и рекомендуется применять, когда $\|A\|$ много меньше $\|N\|^2$.

Процедура, определяющая смежность двух узлов $X$ и $Y$ в графе $G$, представленном списком смежностей, приведена ниже.

```
adjacent(X, Y, [[X | L] | G]):- member(Y, L), !.
adjacent(X, Y, [_ | G]):- adjacent(X, Y, G).
```

Процедура, описывающая связность в графе, имеет следующий вид:

```
connect(X, Y, G):-     connect1(X, [Y], G).

connect1(X, [X | _], _).
connect1(X, [Y | L], G):-      adjacent(Z, Y, G),
                               not(member(Z, L)),
                               connect1(X, [Z,Y | L], G).
```

Ориентированный граф можно задать с помощью *базы данных*, в которой каждое ребро задается в виде отдельного факта следующего вида:

```
edge(a, b).
```

Указанный факт входит в программу, если в графе существует ребро, ведущее из вершины ***a*** в вершину ***b***.

Коммутативное отношение для представления неориентированного графа можно задать следующим образом:

```
arc(X, Y):- edge(X, Y).
arc(X, Y):- edge(Y, X).
```

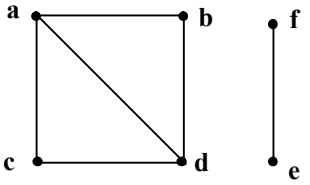
Тогда процедура, описывающая связность в неориентированном графе, будет иметь следующий вид:

```
connected(X, Y):- path(X, Y, [X]).

path(X, Y, _):- arc(X, Y),!.
path(X, Y, H):-  arc(X, Z),   not(member(Z, H)),
                                  path(Z, Y, [Z | H]).
```

Кроме отношения коммутативности **arc** здесь присутствует список просмотренных вершин (история просмотра графа). Этот список анализируется всякий раз, когда программа попадает в очередную вершину графа. Если вершина присутствует в списке, то повторно в него она не записывается, тем самым исключается возможность зацикливания программы.

Расширим возможности программы, заставив находить все возможные пути из одной вершины в другую. Для этого введем в отношение ***path*** еще один аргумент – список пройденных вершин.

Ниже представлен граф и полная программа представления и нахождения пути между двумя вершинами данного графа.




```
edge (a, b).   edge (a, c).        edge (a, d).        edge (d, c).
edge (d, b).   edge (f, e).
arc (X, Y):- edge(X, Y).
arc (X, Y):- edge(Y, X).

path(X, Y, _, []):-arc (X, Y).
path(X, Y, H, [Z | W]):- arc (X, Z), not(member(Z, H)),
```

```
                                        path(Z, Y, [Z | H], W).

member(X, [X| _]).
member(X, [ _ | Y]):- member(X, Y).

gl( X, Y):- findall(W, path(X, Y, [X, Y], W), L), write(L).
```

Исходя из определения графа, его можно представить как пару множеств – вершин и ребер. Множество вершин задается здесь с помощью простого списка. Для записи ребра будем использовать функтор ***tr***, аргументами которого являются пара вершин. Для объединения двух множеств можно использовать функтор ***graph***. Приведенный выше граф в принятых соглашениях будет записан следующим образом:

```
G =graph( [a, b, c, d, e, f], [tr(a, b), tr(a, c), tr(a, d),  tr(d, c),
          tr(d, b), tr(f, e)]).
```

Если в графе каждая из вершин соединена хотя бы с одной другой вершиной, то такой связный граф полностью может быть описан множеством своих ребер. Это еще один способ представления графа, здесь для записи ребра будем также использовать функтор ***tr***. Например, граф, представленный выше, может быть записан следующим образом:

```
G = [tr(a, b), tr(a, c), tr(a, d),  tr(d, c), tr(d, b), tr(f, e)].
```

Процедура, описывающая связность в неориентированном графе, представленном таким образом, будет иметь следующий вид:

```
connect(X, Y, G):-     connect1(X, [Y], G).

connect1(X, [X|_], _).
connect1(X, [Y|L], G):-        adjacent(Z, Y, G),
                                not(member(Z, L)),
                                connect1(X, [Z,Y|L], G).

adjacent(X, Y, G):- member(tr(X, Y), G).
adjacent(X, Y, G):- member(tr(Y, X), G).
```

Если необходимо конкретно указать путь ***P*** между двумя вершинами ***X*** и ***Y*** данного графа ***G***, то программа, построенная на основе указанных выше отношений, будет иметь следующий вид:

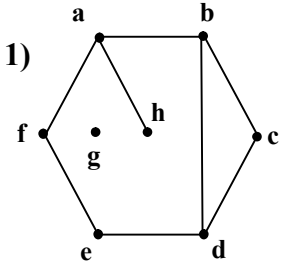
```
path(X,Y,G,P):-         path1(X,[Y],G,P).

path1(X,[X|P],_,[X|P]).
```

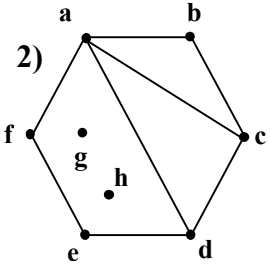
```
path1(X,[Y|L],G,P):- adjacent(Z,Y,G),
                     not(member(Z,L)),
                     path1(X,[Z,Y|L],G,P).
```

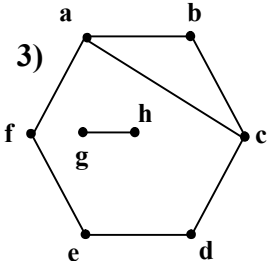
### *Задание*

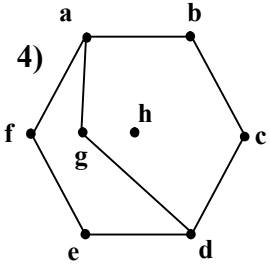
1. Получите вариант задания. Самостоятельно постройте ориентированный граф на основе заданного неориентированного графа.
2. Согласно заданному варианту опишите представления ориентированного и неориентированного графов в виде *списков смежностей* и *базы данных*.
3. Задайте операции поиска пути в нотации Visual Prolog на ориентированном и неориентированном графах, представленных в виде *списков смежностей* и *баз данных*.
4. Исследуйте методы представления графов и поиска пути на ориентированном и неориентированном графе. Сравните эти методы по эффективности.

**Варианты**



# Литература


1. Чень Ч., Ли Р. Математическая логика и автоматическое доказательство теорем. М.: Наука, 1983. — 360 с.
2. Ходашинский И.А. Язык ПРОЛОГ в примерах и задачах. – Томск: Томск. гос. ун-т управл. и радиоэлектроники, 2006. – 280 с. ISBN 5-86889-291-7.
3. Ходашинский И.А. Методы искусственного интеллекта, базы знаний, экспертные системы. Учебное пособие. – Томск: Томск. гос. университет систем управления и радиоэлектроники, 2002. – 139 с.
4. Представление и использование знаний. / Под ред. Х. Уэно, М. Исудзука. – М.: Мир, 1989. – 220 с.
5. Ин. Ц., Соломон Д. Использование Турбо-Пролога. – М: Мир, 1993. – 608 с.
6. Советов, Борис Яковлевич. Представление знаний в информационных системах: учебник для вузов / Б. Я. Советов, В. В. Цехановский, В. Д. Чертовской. - М.: Академия, 2011. - 144 с.
7. Шрайнер, Павел. Основы программирования на языке Пролог: Курс лекций. Учебное пособие / П. А. Шрайнер; Интернет-Университет Информационных Технологий. - М.: Интернет-Университет Информационных Технологий, 2005. - 172 с.