

**ТЕОРЕТИЧЕСКИЕ ДОМАШНИЕ ЗАДАНИЯ**  
Математическая логика, ИТМО, М3232-М3239, осень 2023 года

**Задание №1. Знакомство с исчислением высказываний.**

Справочное изложение теории, частично разобранный на лекции.

**Определение 1.** Аксиомой является любая формула исчисления высказываний, которая может быть получена из следующих схем аксиом:

- (1)  $\alpha \rightarrow \beta \rightarrow \alpha$
- (2)  $(\alpha \rightarrow \beta) \rightarrow (\alpha \rightarrow \beta \rightarrow \gamma) \rightarrow (\alpha \rightarrow \gamma)$
- (3)  $\alpha \rightarrow \beta \rightarrow \alpha \& \beta$
- (4)  $\alpha \& \beta \rightarrow \alpha$
- (5)  $\alpha \& \beta \rightarrow \beta$
- (6)  $\alpha \rightarrow \alpha \vee \beta$
- (7)  $\beta \rightarrow \alpha \vee \beta$
- (8)  $(\alpha \rightarrow \gamma) \rightarrow (\beta \rightarrow \gamma) \rightarrow (\alpha \vee \beta \rightarrow \gamma)$
- (9)  $(\alpha \rightarrow \beta) \rightarrow (\alpha \rightarrow \neg \beta) \rightarrow \neg \alpha$
- (10)  $\neg \neg \alpha \rightarrow \alpha$

**Определение 2.** Выводом из гипотез  $\gamma_1, \dots, \gamma_n$  назовём конечную непустую последовательность высказываний  $\delta_1, \dots, \delta_t$ , для каждого из которых выполнено хотя бы что-то из списка:

1. высказывание является аксиомой;
2. высказывание получается из предыдущих по правилу *Modus Ponens* (то есть, для высказывания  $\delta_i$  найдутся такие  $\delta_j$  и  $\delta_k$ , что  $j, k < i$  и  $\delta_k \equiv \delta_j \rightarrow \delta_i$ );
3. высказывание является гипотезой (то есть, является одной из формул  $\gamma_1, \dots, \gamma_n$ ).

**Определение 3.** Будем говорить, что формула  $\alpha$  выводится (доказывается) из гипотез  $\gamma_1, \dots, \gamma_n$  (и записывать это как  $\gamma_1, \dots, \gamma_n \vdash \alpha$ ), если существует такой вывод из гипотез  $\gamma_1, \dots, \gamma_n$ , что последней формулой которого является формула  $\alpha$ .

Заметим, что доказательство формулы  $\alpha$  — это вывод формулы  $\alpha$  из пустого множества гипотез.

При решении заданий вам может потребоваться теорема о дедукции (будет доказана на второй лекции):

**Теорема 1.**  $\gamma_1, \dots, \gamma_n, \alpha \vdash \beta$  тогда и только тогда, когда  $\gamma_1, \dots, \gamma_n \vdash \alpha \rightarrow \beta$ .

Пример использования: пусть необходимо доказать  $\vdash A \rightarrow A$  — то есть доказать существование вывода формулы  $A \rightarrow A$  (заметьте, так поставленное условие не требует этот вывод предъявлять, только доказать его существование). Тогда заметим, что последовательность из одной формулы  $A$  доказывает  $A \vdash A$ . Далее, по теореме о дедукции, отсюда следует и  $\vdash A \rightarrow A$  (то есть, вывода формулы  $A \rightarrow A$ , не использующего гипотезы).

1. Докажите:

- (a)  $\vdash (A \rightarrow A \rightarrow B) \rightarrow (A \rightarrow B)$
- (b)  $\vdash \neg(A \& \neg A)$
- (c)  $\vdash A \& B \rightarrow B \& A$
- (d)  $\vdash A \vee B \rightarrow B \vee A$
- (e)  $A \& \neg A \vdash B$

2. Докажите:

- (a)  $\vdash A \rightarrow \neg \neg A$
- (b)  $\neg A, B \vdash \neg(A \& B)$
- (c)  $\neg A, \neg B \vdash \neg(A \vee B)$
- (d)  $A, \neg B \vdash \neg(A \rightarrow B)$
- (e)  $\neg A, B \vdash A \rightarrow B$

3. Докажите:

- (a)  $\vdash (A \rightarrow B) \rightarrow (B \rightarrow C) \rightarrow (A \rightarrow C)$
  - (b)  $\vdash (A \rightarrow B) \rightarrow (\neg B \rightarrow \neg A)$  (правило контрапозиции)
  - (c)  $\vdash \neg(\neg A \& \neg B) \rightarrow (A \vee B)$  (вариант I закона де Моргана)
  - (d)  $\vdash (\neg A \vee \neg B) \rightarrow \neg(A \& B)$  (II закон де Моргана)
  - (e)  $\vdash (A \rightarrow B) \rightarrow (\neg A \vee B)$
  - (f)  $\vdash A \& B \rightarrow A \vee B$
  - (g)  $\vdash ((A \rightarrow B) \rightarrow A) \rightarrow A$  (закон Пирса)
  - (h)  $\vdash A \vee \neg A$
  - (i)  $\vdash (A \& B \rightarrow C) \rightarrow (A \rightarrow B \rightarrow C)$
  - (j)  $\vdash (A \rightarrow B \rightarrow C) \rightarrow (A \& B \rightarrow C)$
  - (k)  $\vdash (A \rightarrow B) \vee (B \rightarrow A)$
  - (l)  $\vdash (A \rightarrow B) \vee (B \rightarrow C) \vee (C \rightarrow A)$
4. Даны высказывания  $\alpha$  и  $\beta$ , причём  $\vdash \alpha \rightarrow \beta$  и  $\not\vdash \beta \rightarrow \alpha$ . Укажите способ построения высказывания  $\gamma$ , такого, что  $\vdash \alpha \rightarrow \gamma$  и  $\vdash \gamma \rightarrow \beta$ , причём  $\not\vdash \gamma \rightarrow \alpha$  и  $\not\vdash \beta \rightarrow \gamma$ .
5. Покажите, что если  $\alpha \vdash \beta$  и  $\neg \alpha \vdash \beta$ , то  $\vdash \beta$ .

## Задание №2. Теоремы об исчислении высказываний. Знакомство с интуиционистским исчислением высказываний.

- (только для очной практики) на память приведите греческий алфавит — запишите на доске в алфавитном порядке все большие и маленькие греческие буквы и назовите их.
- Давайте вспомним, что импликация правоассоциативна:  $\alpha \rightarrow \beta \rightarrow \gamma \equiv \alpha \rightarrow (\beta \rightarrow \gamma)$ . Но рассмотрим иную расстановку скобок:  $(\alpha \rightarrow \beta) \rightarrow \gamma$ . Возможно ли доказать логическое следствие между этими вариантами расстановки скобок — и каково его направление?
- Покажите, что в классическом исчислении высказываний  $\Gamma \models \alpha$  влечёт  $\Gamma \vdash \alpha$ .
- Покажите, что в классическом исчислении высказываний  $\Gamma \vdash \alpha$  влечёт  $\Gamma \models \alpha$ .
- Возможно ли, что какая-то из аксиом задаётся двумя разными схемами аксиом? Опишите все возможные коллизии, если они есть. Ответ обоснуйте (да, тут потребуются доказательства по индукции).
- Заметим, что можно вместо отрицания ввести в исчисление ложь. Рассмотрим *исчисление высказываний с ложью*. В этом языке будет отсутствовать одноместная связка ( $\neg$ ), вместо неё будет присутствовать нульместная связка «ложь» ( $\perp$ ), а 9 и 10 схемы аксиом будут заменены на одну схему:

$$(9_{\perp}) \quad ((\alpha \rightarrow \perp) \rightarrow \perp) \rightarrow \alpha$$

Будем записывать доказуемость в новом исчислении как  $\vdash_{\perp} \alpha$ , а доказуемость в исчислении высказываний с отрицанием как  $\vdash_{\neg} \beta$ .

Определим операцию трансляции между языками обычного исчисления высказываний и исчисления с ложью как операции рекурсивной замены  $\perp := A \& \neg A$  и  $\neg \alpha := \alpha \rightarrow \perp$  (и обозначим их как  $|\varphi|_{\neg}$  и  $|\psi|_{\perp}$  соответственно), то:

- (a)  $\vdash_{\perp} \alpha$  влечёт  $\vdash_{\neg} |\alpha|_{\neg}$
- (b)  $\vdash_{\neg} \alpha$  влечёт  $\vdash_{\perp} |\alpha|_{\perp}$

- Изоморфизм Карри-Ховарда — соответствие между логическими исчислениями (например, исчислением высказываний), с одной стороны, и языками программирования, с другой. А именно, можно заметить, что программа соответствует доказательству, тип программы — логическому высказыванию. Связки (как составные части логического высказывания) соответствуют определённым типовым конструкциям: функция — импликация, конъюнкция — упорядоченной паре, дизъюнкция — алгебраическому типу (`std::variant` и т.п.). Атомарным высказываниям мы сопоставим элементарные типы. Понятие же доказуемости превращается в *обитаемость* типа. Например, доказать обитаемость типа `int` возможно, предъявив значение этого типа: 5.

Функция `A id(A x) { return x; }` доказывает  $A \rightarrow A$ , а функция

```
std::pair<A,B> swap(std::pair<B,A> x) { return std::pair(x.second, x.first); }
```

доказывает  $B \& A \rightarrow A \& B$ . В самом деле, данные функции являются элементами соответствующих типов, поэтому их можно понимать как доказательства соответствующих типов логических выражений.

Ложь — это необитаемый тип; тип, не имеющий значений. В некоторых языках такие типы можно выписать явно. Например, в Хаскеле можно построить алгебраический тип без конструкторов:

```
data False
main = do print "Hi"
```

В других (например, в C++) эти значения можно симитировать. Например, в одних случаях сделать параметром темплейта. Тогда, если мы никаких ограничений на этот параметр не делаем, кто-то мог бы подставить и необитаемый тип вместо этого параметра:

```
template <class Bot>
Bot (*contraposition (A a)) (A a, B b, Bot (*neg_b) (B));
```

В самом деле,  $(A \rightarrow B) \rightarrow ((B \rightarrow \perp) \rightarrow (A \rightarrow \perp))$  есть частный случай  $(A \rightarrow B) \rightarrow ((B \rightarrow \alpha) \rightarrow (A \rightarrow \alpha))$ , который тоже можно доказать.

В некоторых случаях можно воспользоваться конструкцией, не возвращающей управления, которая *понятна компилятору*. Например, можно так задать правило удаления лжи ( $\perp \rightarrow A$ ):

```
template <class Bot>
A remove_bot(Bot x) { throw x; }

int a = remove_bot<int> (...);
char* b = remove_bot<char*> (...);
char(*c)() = remove_bot<char(*)()> (...);
```

В завершение теоретической части заметим, что

- логика, которая получится, если мы будем играть в эту игру честно — это уже будет не классическая логика; для неё не будут справедливы все схемы аксиом, 10 схема будет нарушаться;
- большинство языков программирования противоречивы в смысле логической теории; в частности, там можно доказать ложь. Но для того, чтобы это получилось, вам обычно требуется использовать либо инструменты обхода ограничений типовой системы (например, явные приведения типов), либо конструкции, не возвращающие управления: бесконечная рекурсия, исключения и т.п.

Докажите следующие утверждения, написав соответствующую программу на выбранном вами языке программирования, не используя противоречивости его типовой системы (кроме последнего задания). В случае C++ можно также использовать правило удаления лжи, указанное выше; для других языков при необходимости можно выделить какое-то похожее правило:

- $A \rightarrow B \rightarrow A$
- $A \& B \rightarrow A \vee B$
- $(A \& (B \vee C)) \rightarrow ((A \& B) \vee (A \& C))$
- $(A \rightarrow C) \& (B \rightarrow C) \& (A \vee B) \rightarrow C$
- $(B \vee C \rightarrow A) \rightarrow (B \rightarrow A) \& (C \rightarrow A)$
- $(A \rightarrow B) \rightarrow (\neg B \rightarrow \neg A)$
- $((A \rightarrow B) \rightarrow C) \rightarrow (A \rightarrow (B \rightarrow C))$
- $(A \rightarrow B) \& (A \rightarrow \neg B) \rightarrow \neg A$
- Одно из двух утверждений:  $(A \rightarrow B) \rightarrow \neg A \vee B$  или  $\neg A \vee B \rightarrow (A \rightarrow B)$ . Сразу заметим, что оставшееся утверждение доказать без использования противоречивости языка не получится.
- $\perp$  (любым доступным в языке способом)

Для зачёта по пункту условия требуется написать код программы и продемонстрировать его работу на компьютере. Если вы желаете получить дополнительные 0.5 балла за оформление в Тех-е, вам потребуется оформить в Тех-е исходный код программы (подсказка: для языков программирования могут существовать специальные пакеты для красивого оформления кода).