Теоретические домашние задания

Математическая логика, ИТМО, M3232-M3239, осень 2023 года

Задание №1. Знакомство с исчислением высказываний.

Справочное изложение теории, частично разобранной на лекции.

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

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

Определение 2. *Выводом из гипотез $\gamma_1, \ldots, \gamma_n$ назовём конечную непустую последовательность высказываний $\delta_1, \ldots, \delta_t$, 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 δ_i найдутся такие δ_j и δ_k, что $j, k < i$ и $\delta_k \equiv \delta_j \to \delta_i$);*
3. *высказывание является гипотезой (то есть, является одной из формул $\gamma_1, \ldots, \gamma_n$).*

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

Заметим, что доказательство формулы α — это вывод формулы α из пустого множества гипотез.

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

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

Пример использования: пусть необходимо доказать $\vdash A \to A$ — то есть доказать существование вывода формулы $A \to A$ (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 $\vdash A \to A$ (то есть, вывода формулы $A \to A$, не использующего гипотезы).

1. Докажите:
   (a) $\vdash (A \to A \to B) \to (A \to B)$
   (b) $\vdash \neg(A \& \neg A)$
   (c) $\vdash A \& B \to B \& A$
   (d) $\vdash A \vee B \to B \vee A$
   (e) $A \& \neg A \vdash B$
2. Докажите:
   (a) $\vdash A \to \neg\neg A$
   (b) $\neg A, B \vdash \neg(A \& B)$
   (c) $\neg A, \neg B \vdash \neg(A \vee B)$
   (d) $A, \neg B \vdash \neg(A \to B)$
   (e) $\neg A, B \vdash A \to B$
3. Докажите:

(a) $\vdash (A \to B) \to (B \to C) \to (A \to C)$

(b) $\vdash (A \to B) \to (\neg B \to \neg A)$ *(правило контрапозиции)*

(c) $\vdash \neg(\neg A \,\&\, \neg B) \to (A \vee B)$ *(вариант I закона де Моргана)*

(d) $\vdash (\neg A \vee \neg B) \to \neg(A \,\&\, B)$ *(II закон де Моргана)*

(e) $\vdash (A \to B) \to (\neg A \vee B)$

(f) $\vdash A \,\&\, B \to A \vee B$

(g) $\vdash ((A \to B) \to A) \to A$ *(закон Пирса)*

(h) $\vdash A \vee \neg A$

(i) $\vdash (A \,\&\, B \to C) \to (A \to B \to C)$

(j) $\vdash (A \to B \to C) \to (A \,\&\, B \to C)$

(k) $\vdash (A \to B) \vee (B \to A)$

(l) $\vdash (A \to B) \vee (B \to C) \vee (C \to A)$

4. Даны высказывания α и β, причём $\vdash \alpha \to \beta$ и $\nvdash \beta \to \alpha$. Укажите способ построения высказывания γ, такого, что $\vdash \alpha \to \gamma$ и $\vdash \gamma \to \beta$, причём $\nvdash \gamma \to \alpha$ и $\nvdash \beta \to \gamma$.

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.

1. (только для очной практики) На память приведите греческий алфавит — запишите на доске в алфавитном порядке все большие и маленькие греческие буквы и назовите их.

2. Давайте вспомним, что импликация правоассоциативна: $\alpha \to \beta \to \gamma \equiv \alpha \to (\beta \to \gamma)$. Но рассмотрим иную расстановку скобок: $(\alpha \to \beta) \to \gamma$. Возможно ли доказать логическое следствие между этими вариантами расстановки скобок — и каково его направление?

3. Покажите, что в классическом исчислении высказываний $\Gamma \models \alpha$ влечёт $\Gamma \vdash \alpha$.

4. Покажите, что в классическом исчислении высказываний $\Gamma \vdash \alpha$ влечёт $\Gamma \models \alpha$.

5. Возможно ли, что какая-то из аксиом задаётся двумя разными схемами аксиом? Опишите все возможные коллизии, если они есть. Ответ обоснуйте (да, тут потребуется доказательство по индукции).

6. 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:

   $(9_\perp) \quad ((\alpha \to \perp) \to \perp) \to \alpha$

   Будем записывать доказуемость в новом исчислении как $\vdash_\perp \alpha$, а доказуемость в исчислении высказываний с отрицанием как $\vdash_\neg \beta$. Также определим операцию трансляции между языками обычного исчисления высказываний и исчисления с ложью как операции рекурсивной замены $\perp := A \,\&\, \neg A$ и $\neg\alpha := \alpha \to \perp$ (и обозначим их как $|\varphi|_\neg$ и $|\psi|_\perp$ соответственно).

   Докажите:

   (a) $\vdash_\perp \alpha$ влечёт $\vdash_\neg |\alpha|_\neg$

   (b) $\vdash_\neg \alpha$ влечёт $\vdash_\perp |\alpha|_\perp$

7. 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 — импликации, 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 $A \to A$, а функция

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

доказывает $B \,\&\, A \to A \,\&\, B$. В самом деле, данные функции являются элементами соответствующих типов, поэтому их можно понимать как доказательства соответствующих типам 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 сымитировать. 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, $(A \to B) \to ((B \to \perp) \to (A \to \perp))$ есть частный случай $(A \to B) \to ((B \to \alpha) \to (A \to \alpha))$, который тоже можно доказать.

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

```
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 заверешние 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) $A \to B \to A$

(b) $A \,\&\, B \to A \vee B$

(c) $(A \,\&\, (B \vee C)) \to ((A \,\&\, B) \vee (A \,\&\, C))$

(d) $(A \to C) \,\&\, (B \to C) \,\&\, (A \vee B) \to C$

(e) $(B \vee C \to A) \to (B \to A) \,\&\, (C \to A)$

(f) $(A \to B) \to (\neg B \to \neg A)$

(g) $((A \to B) \to C) \to (A \to (B \to C))$

(h) $(A \to B) \,\&\, (A \to \neg B) \to \neg A$

(i) $(A \to B \to C) \to ((A \,\&\, B) \to C)$

(j) $A \vee B \to \neg(\neg A \,\&\, \neg B)$

(k) Одно из двух утверждений: $(A \to B) \to \neg A \vee B$ или $\neg A \vee B \to (A \to B)$. Сразу заметим, что оставшееся утверждение доказать без использования противоречивости языка не получится.

(l) $\perp$ (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 Tex-е, вам потребуется оформить в Tex-е исходный код программы (подсказка: для языков программирования могут существовать специальные пакеты для красивого оформления кода).