

Домашние задания по курсу «Математическая логика»

ИТМО, группы М3234..М3239
Весна 2018 г.

Общие замечания

Для всех программ кодировка входных и выходных файлов должна быть UTF8. Задания подаются в систему Яндекс.контекст, подробные описания — по ссылке из README.md. Для компиляции решения требуется использования мэйкфайлов, краткое описание принципов построения мэйкфайлов находится в файле make.pdf из данного репозитория.

Задача 0. Разбор выражения

Стоимость: 0 баллов, решение на Ocaml или Haskell: 0 баллов

Данная задача разобрана, решения её приведены (см. README.md), однако, мы крайне рекомендуем написать своё её решение по двум причинам: (а) разбор высказываний можно будет переиспользовать в других задачах; (б) можно протестировать среду исполнения на Яндексе.

На вход программе (в файле `input.txt`) подаётся выражение в следующей грамматике:

$$\begin{aligned}\langle \text{файл} \rangle &::= \langle \text{выражение} \rangle \\ \langle \text{выражение} \rangle &::= \langle \text{дизъюнкция} \rangle \mid \langle \text{дизъюнкция} \rangle \text{'->'} \langle \text{выражение} \rangle \\ \langle \text{дизъюнкция} \rangle &::= \langle \text{конъюнкция} \rangle \mid \langle \text{дизъюнкция} \rangle \text{'|'} \langle \text{конъюнкция} \rangle \\ \langle \text{конъюнкция} \rangle &::= \langle \text{отрицание} \rangle \mid \langle \text{конъюнкция} \rangle \text{'\&'} \langle \text{отрицание} \rangle \\ \langle \text{отрицание} \rangle &::= (\text{'A'} \dots \text{'Z'}) \{ (\text{'A'} \dots \text{'Z'} \mid \text{'0'} \dots \text{'9'})^* \mid \text{'!'} \langle \text{отрицание} \rangle \mid (\text{'('} \langle \text{выражение} \rangle \text{'})'}\end{aligned}$$

Пробелы, символы табуляции и переноса строки должны игнорироваться. Символ `'|'` имеет ASCII-код 124₁₀.

Написать программу, разбирающую выражение и строящую его дерево разбора, и выводящую полученное дерево в файл `output.txt` в следующей грамматике.

$$\begin{aligned}\langle \text{файл} \rangle &::= \langle \text{вершина} \rangle \\ \langle \text{вершина} \rangle &::= (\text{'('} \langle \text{знак} \rangle \text{'}, \langle \text{вершина} \rangle \text{'}, \langle \text{вершина} \rangle \text{'})'} \\ &\quad \mid (\text{'('} \langle \text{вершина} \rangle \text{'})'} \\ &\quad \mid (\text{'A'} \dots \text{'Z'}) \{ (\text{'A'} \dots \text{'Z'} \mid \text{'0'} \dots \text{'9'})^* \} \\ \langle \text{знак} \rangle &::= \text{'\&'} \mid \text{'|'} \mid \text{'->'}\end{aligned}$$

Пример входного файла:

P->!QQ->!R10&S|!T&U&V

Выходной файл для данного входного файла:

(->,P,(->,(!QQ),(|,(&,(!R10),S),(&,(&,(!T),U),V))))

Задача 1. Проверка вывода

ДЕДЛАЙН: 23:59, 8 апреля

Стоимость: 7 баллов, решение на Ocaml или Haskell: 9 баллов

Написать программу, проверяющую вывод $\gamma_1, \dots, \gamma_n \vdash \alpha$ в исчислении высказываний на корректность. Входной файл соответствует следующей грамматике, нетерминал $\langle \text{выражение} \rangle$ определён в грамматике из задачи 0:

$$\begin{aligned}\langle \text{файл} \rangle &::= \langle \text{заголовок} \rangle \langle 'n' \{ \langle \text{выражение} \rangle \langle 'n' \}^* \\ \langle \text{заголовок} \rangle &::= [\langle \text{выражение} \rangle \{ \langle ' \rangle \langle \text{выражение} \rangle \}^*] \langle ' - ' \rangle \langle \text{выражение} \rangle\end{aligned}$$

В первой строке входного файла (заголовок) перечислены предположения γ_i (этот список может быть пустым) и доказываемое утверждение α . В последующих строках указаны формулы, составляющие вывод формулы α . Пробелы, символы табуляции и возврата каретки (ASCII-код 13₁₀) должны игнорироваться. Символ ' | ' имеет ASCII-код 124₁₀.

Результатом работы программы должен быть файл с проаннотированным текстом доказательства, где каждая строка — соответствующая строка из вывода, расширенная в соответствии с грамматикой:

$$\begin{aligned}\langle \text{строка} \rangle &::= \langle ' \langle \text{номер} \rangle ' \rangle \langle \text{выражение} \rangle \langle ' \langle \text{аннотация} \rangle ' \rangle \\ \langle \text{аннотация} \rangle &::= \langle \text{Сх. акс.} \rangle \langle \text{номер} \rangle \\ &| \langle \text{Предп.} \rangle \langle \text{номер} \rangle \\ &| \langle \text{М.Р.} \rangle \langle \text{номер} \rangle \langle ' , ' \rangle \langle \text{номер} \rangle \\ &| \langle \text{Не доказано} \rangle \\ \langle \text{номер} \rangle &::= \{ \langle ' 0' \dots ' 9' \rangle \}^+\end{aligned}$$

Выражение не должно содержать пробелов, номер от выражения и выражение от аннотации должны отделяться одним пробелом. Выражения в доказательстве должны нумероваться подряд натуральными числами с 1. Если выражение δ_n получено из δ_i и δ_j , где $\delta_j \equiv \delta_i \rightarrow \delta_n$ путём применения правила Modus Ponens, то аннотация должна выглядеть как 'М.Р. i, j ', обратный порядок номеров не допускается.

Ограничения

Количество строк в файле не превосходит 52000.

Размер файла не превосходит 10 мегабайт.

Пример 1:

Входной файл:

A, B | -A & B
A
B
A → B → A & B
B → A & B
A & B

Выходной файл:

(1) A (Предп. 1)
(2) B (Предп. 2)
(3) (A → (B → (A & B))) (Сх. акс. 3)
(4) (B → (A & B)) (М.Р. 3, 1)
(5) (A & B) (М.Р. 4, 2)

Пример 2:

Входной файл:

A, B | -A & B
A
B
(A → (B → (A & B)))
(B → (A & B))
(A → A)
(A & B)

Выходной файл:

(1) A (Предп. 1)
(2) B (Предп. 2)
(3) (A → (B → (A & B))) (Сх. акс. 3)
(4) (B → (A & B)) (М.Р. 3, 1)
(5) (A → A) (Не доказано)
(6) (A & B) (М.Р. 4, 2)

Пример 3:

Входной файл:

```
| -A->A
(A->A->A)->(A->(A->A)->A)->(A->A)
(A->A->A)
(A->(A->A)->A)
(A->(A->A)->A)->(A->A)
A->A
```

Выходной файл:

```
(1) (A->A->A)->(A->(A->A)->A)->A->A (Сх. акс. 2)
(2) A->A->A (Сх. акс. 1)
(3) A->(A->A)->A (Сх. акс. 1)
(4) (A->(A->A)->A)->A->A (М.Р. 1, 2)
(5) A->A (М.Р. 4, 3)
```

Пример 4:

Входной файл:

```
| -B
A->B
A
B
```

Выходной файл:

```
(1) (A->B) (Не доказано)
(2) A (Не доказано)
(3) B (М.Р. 1, 2)
```

Задача 2. Теорема о дедукции

ДЕДЛАЙН: 23:59, 15 апреля

Стоимость: 4 балла, решение на Ocaml или Haskell: 6 баллов

Написать программу, преобразующую вывод $\Gamma, \alpha \vdash \beta$ в вывод $\Gamma \vdash \alpha \rightarrow \beta$. Входной файл удовлетворяет грамматике из предыдущего задания, в заголовке обязательно должно присутствовать как минимум одно предположение.

Результатом работы программы должен быть текст, содержащий преобразованный вывод. Формат выходного файла совпадает с форматом входного файла. Вы можете предполагать, что входной файл содержит корректный вывод требуемой формулы.

Ограничения

Небольшие.

Пример 1:

Входной файл:

```
A, A | -A
A
```

Выходной файл:

```
A | -A->A
A->A->A
A
A->A
```

Пример 2:

Входной файл:

```
A | -B->A
A->B->A
A
B->A
```

Выходной файл:

```
| -A->B->A
A->B->A
```

Задача 3. Теорема о полноте исчисления высказываний

ДЕДЛАЙН: 23:59, 2 мая

Стоимость: 10 баллов, решение на Ocaml или Haskell: 13 баллов

Будем называть формулу классического исчисления высказываний ϕ *логическим следствием* формул $\gamma_1, \dots, \gamma_n$ (и записывать это как $\gamma_1, \dots, \gamma_n \models \phi$), если для любой оценки пропозициональных переменных M , такой, что $\llbracket \gamma_k \rrbracket_M = \text{И}$, выполнено $\llbracket \phi \rrbracket_M = \text{И}$. Иными словами, формула ϕ истинна всегда, когда истинны все γ_k .

Написать программу, проверяющую $\gamma_1, \dots, \gamma_n \models \phi$ и строящую доказательство $\gamma_1, \dots, \gamma_n \vdash \phi$ в случае успешной проверки, либо строящую контрпример в случае неуспеха.

Входной файл состоит из единственной строки:

$$[\{ \langle \text{выражение} \rangle ' , ' \}^* \langle \text{выражение} \rangle] ' | = ' \langle \text{выражение} \rangle$$

Выходной файл должен либо содержать доказательство высказывания (в формате входного файла из первого задания), либо содержать фразу, удовлетворяющую грамматике:

$$\begin{aligned} \langle \text{строка} \rangle &::= \text{'Высказывание ложно при ' } \langle \text{назначение} \rangle \{ ' , ' \langle \text{назначение} \rangle \}^* \\ \langle \text{назначение} \rangle &::= \langle \text{переменная} \rangle ' = ' (' \text{И} ' | ' \text{Л} ') \end{aligned}$$

Ограничения

Количество связок не превосходит 12 (например в выражении $A \wedge B \vdash A \vee B$ — 2 связки).
Количество различных переменных не превосходит 5.

Пример 1:

Входной файл:

$| = !A \& !B$

Выходной файл:

Высказывание ложно при $A=\text{И}, B=\text{Л}$

Пример 2:

Входной файл:

$B, W | = A \rightarrow B$

Выходной файл:

$B, W | = A \rightarrow B$
 $B \rightarrow A \rightarrow B$
 B
 $A \rightarrow B$

Задача 4. Решётки

ДЕДЛАЙН: 23:59, 20 мая

Стоимость: 6 баллов, решение на *Ocaml* или *Haskell*: 8 баллов

По заданному на вход вашей программе графу требуется установить, задаёт ли его рефлексивное и транзитивное замыкание решётку, а также, является ли она дистрибутивной, имплективной решёткой, булевой алгеброй. Гарантируется, что рефлексивное и транзитивное замыкание графа задаёт частичный порядок.

Вершины графа мы предполагаем занумерованными числами от 1 до v . Входной файл в первой строке содержит число вершин v , после чего идёт ещё v строк, по строке для каждой вершины: вершине i соответствует строка номер $i + 1$ входного файла. В каждой такой строке через пробел перечислены все такие вершины i_k , что $i \sqsubseteq i_k$. Гарантируется, что все эти строки содержат хотя бы по одной вершине.

Выходной файл должен соответствовать следующей грамматике:

$$\begin{aligned} \langle \text{выходной файл} \rangle &::= \text{'Операция ' + ' не определена: ' } \langle \text{вершина} \rangle ' + ' \langle \text{вершина} \rangle \\ &\quad | \text{'Операция ' * ' не определена: ' } \langle \text{вершина} \rangle ' * ' \langle \text{вершина} \rangle \\ &\quad | \text{'Нарушается дистрибутивность: ' } \langle \text{вершина} \rangle ' * (' \langle \text{вершина} \rangle ' + ' \langle \text{вершина} \rangle ') ' \end{aligned}$$

	‘Операция ’->’ не определена: ’⟨вершина⟩’->’⟨вершина⟩
	‘Не булева алгебра: ’⟨вершина⟩’+~’⟨вершина⟩
	‘Булева алгебра’

Вам следует находить самое слабое свойство (например, если граф не является решёткой, то указание на нарушение дистрибутивности будет ошибкой). *Для данной задачи* будем считать, что чем ниже свойство указано в данной грамматике, тем оно сильнее.

Ограничения

Количество вершин в графе не превосходит 100.

Пример 1:

Входной файл:

```
5
2 3 4
5
5
5
5
```

Выходной файл:

Нарушается дистрибутивность: $2*(3+4)$