

## **Аннотация**

Тут будет аннотация

## Содержание

<b>1</b>	<b>Введение</b>	<b>4</b>
<b>2</b>	<b>Описание подхода</b>	<b>5</b>
2.1	Подготовительный этап . . . . .	5
2.2	Обработка предложения . . . . .	5
2.2.1	Упрощение предложения . . . . .	6
	<b>Список литературы</b>	<b>10</b>
	<b>ПРИЛОЖЕНИЕ А. Система правил для анализа словосочетаний из двух слов</b>	<b>11</b>

## **1. Введение**

Тут будет введение

## 2. Описание подхода

### 2.1. Подготовительный этап

На вход программе подаётся предложение, состоящее из существительных, местоимений, личных глаголов или (и) инфинитивов (с, возможно, перечислением инфинитивов или личных глаголов с зависимыми словами, принадлежащим указанным частям речи).

Полученное предложение передаётся функции `space()`, которая преобразует считанную строку в список. Механизм её работы описывает алгоритм 1.

---

Алгоритм 1 – Предварительная обработка входных данных

---

```
1: function SPACE(str1)
2:   str1 ← str1.lower()      ▷ Приводим полученную строку к нижнему регистру
3:   str2 ← «»                ▷ В этой переменной будет храниться преобразованная строка
4:   l ← len(str1)
5:   for from i = 0 to l – 1 do
6:     if str1[i] ∈ {«.», «,», «.»} then
7:       str2 ← str2 + « »
8:     end if
9:     str2 ← str2 + str1[i]
10:  end for
11:  if str2[len(str2) – 1] = « » then      ▷ Если последним элементом полученного
    списка оказался пробел
12:    str2 ← str2[ : len(str2) – 1]          ▷ Отбрасываем этот пробел
13:  end if
14:  return str2.split()                    ▷ Возвращаем список, полученный из строки str2
    разбиением её по пробелам
15: end function
```

---

Таким образом, функция `space()` возвращает список, состоящий из слов и знаков препинания исходной строки.

### 2.2. Обработка предложения

Итак, как было сказано выше, проверка согласования единственного и множественного числа в русском языке — процесс сложный: нужно учесть много критериев.

В основе предложенного нами подхода лежит гипотеза, согласно которой одно и то же предложение являться и не являться ошибочным одновременно с точки зрения согласования единственного и множественного числа не может.

Также считаем, что в предложении нет орфографических, пунктуационных и

др. ошибок, поскольку данная задача была успешно решена, например, компанией LanguageTooler GmbH [2].

Нами было принято решение декомпозировать задачу.

Для начала (при наличии перечислений инфинитивов или личных личных глаголов) предложение упрощается: перечисление мы заменяем на инфинитив или личный глагол соответственно (параллельно проверяя, что внутри заменяемой части нет ошибок в согласовании единственного и множественного числа). Если же перечисления не обнаружено, сразу переходим к следующему этапу.

Затем проверяем предложение без перечислений при помощи разработанной нами системы правил.

Согласно теореме Гёделя о неполноте, формальная арифметика либо противоречива, либо неполна [1]. Чтобы избежать противоречивости разработанной системы, мы включили лишь те правила, которые встречаются на практике, а не перебрали все возможные комбинации используемых нами параметров.

### 2.2.1. Упрощение предложения

Под упрощением мы будем понимать замену перечисления инфинитивов или личных глаголов одиночным инфинитивом или личным глаголом.

За упрощение предложения отвечает функция `comma()`, которая принимает на вход список, полученный из исходного предложения при помощи функции `space()`, описанной выше; а возвращает список, в виде которого представлено упрощённое предложение. Механизм работы функции `comma()` описывает алгоритм 2.

---

#### Алгоритм 2 – Обработка перечислений

---

1: <b>function</b> COMMA(l)	▷ l — подготовленная строка в виде списка
2: <b>if</b> «и» <b>in</b> l <b>then</b>	
3:     part ← [ ]	▷ Список значений параметра «часть речи» для данного слова (изначально пустой)
4:     left ← (−1)	
5:     right ← (−1)	▷ Левая и правая границы заменяемого участка, изначально инициализируем невозможными значениями: (−1)
6:     llen ← len(l)	▷ Длина исходного списка
7:     id1 ← l.index(«и»)	▷ Записываем индекс «и» в списке

---

Для начала инициализируем переменные, затем находим индекс вхождения «и» в список (при условии, что в списке есть «и»). После этого анализируем слова, находящиеся в окрестности слова «и»:

```

8:      if llen > id1 +1 then
9:          resp1 ← l[id1 +1].[pos, singugar, cow] ▷ Варианты интерпретации слова,
            стоящего за «и»
10:     end if
11:     if llen > id1 +2 then
12:         resp2 ← l[id1 +2].[pos, singugar, cow]
13:     end if
14:     if llen > id1 +3 then
15:         resp3 ← l[id1 +3].[pos, singugar, cow]
16:     end if
17:     if llen > id1 +4 then
18:         resp4 ← l[id1 +4].[pos, singugar, cow]
19:     end if
20:     if id1 -1 ≥ 0 then
21:         resl1 ← l[id1 -1].[pos, singugar, cow]
22:     end if
23:     if id1 -2 ≥ 0 then
24:         resl2 ← l[id1 -2].[pos, singugar, cow]
25:     end if
26:     if id1 -3 ≥ 0 then
27:         resl3 ← l[id1 -3].[pos, singugar, cow]
28:     end if
29:     if id1 -4 ≥ 0 then
30:         resl1 ← l[id1 -4].[pos, singugar, cow]
31:     end if
32:     if id1 -5 ≥ 0 then
33:         resl5 ← l[id1 -5].[pos, singugar, cow]
34:     end if
35:     if id1 -6 ≥ 0 then
36:         resl6 ← l[id1 -6].[pos, singugar, cow]
37:     end if
38:     if id1 +1 < llen then
39:         for from i = 0 to len(resp1)-1 do
40:             if resp1[i][0] = «6» then ▷ Слово оказалось инфинитивом
41:                 part ← «6»
42:                 right ← id1 +1
43:             end if
44:         end for
45:         if right = (-1) then ▷ Если же это не инфинитив
46:             for from i = 0 to len(resp1)-1 do
47:                 if resp1[i][0] = «5» then ▷ Слово оказалось инфинитивом
48:                     right ← id1+1
49:                     part ← «5»
50:                     sng ← l[i][1] ▷ В отличие от инфинитивов для личных

```

ГЛАГОЛОВ ВАЖНО ЧИСЛО

---

Таким образом, в результате исполнения блока 3 будет определено, слова (словосочетания) какой части речи перечисляются (если в предложении присутствует перечисление с союзом «и»).

Заметим, что в случае перечисления с союзом «и» за союзом идёт слово той же части речи, что и остальные перечисляемые слова. Например: «Он хотел **читать** книги, **рисовать** картины и **познавать** тайны мироздания». Легко видеть, что в предложении перечисляются инфинитивы, и в то же время после союза «и» идёт инфинитив «познавать».

Если перечисляются инфинитивы, то упрощение предложения идёт согласно алгоритму 4.

Прежде всего, нужно определить начало левого операнда «и». В зависимости от длины буквосочетания, возможны различные варианты:

1. Словосочетание длины 6. Например, инф. + сущ. + сущ. + сущ. + сущ. + сущ.: «Проводить проверку знаний требований охраны труда».
2. Словосочетание длины 5. Например, инф. + сущ. + сущ. + сущ. + сущ.: «Организовать проверку знаний основ программирования».
3. Словосочетание длины 4. Например, инф. + инф. + сущ. + сущ.: «Пойти спать сном младенца».
4. Словосочетание длины 3. Например, инф. + сущ. + сущ.: «Оценить игру слов».
5. Словосочетание длины 2. Например, инф. + инф.: «Пойти позавтракать».
6. Одиночный инфинитив. Например: «Быть».

---

#### Алгоритм 4 – Продолжение алгоритма 3

---

```

51:                if part = «6» then
52:                if id1-4 ≥ 0 and left = (-1) and «,» ∉ l[id1-4 : id1] then ▷
    Провераем буквосочетания длины 4
53:                for from i = 0 to len(resl4)-1 do
54:                r ← check(l[id1-4 : id1])
55:                if «N» ∈ r then
56:                return [«он», «писали» ]    ▷ Заведомо неверное
    предложение

```

---

```

57:                                     else if «Y» ∈ r then
58:                                     left ← id1-4           ▷ Инициализировали границу
    левого операнда «и»
59:                                     break
60:                                     end if
61:                                     end for
62:                                     end if
63:                                     if id1-3 ≥ 0 and left = -1 and «,» ∉ l[id1-3 : id1] then
64:                                     for from i = 0 to len(resl3)-1 do
65:                                     r ← check(l[id1-3 : id1])
66:                                     if «N» ∈ r then
67:                                     return [«он», «писали» ]
68:                                     else if «Y» ∈ r then
69:                                     end if
70:                                     left ← id1-3
71:                                     break
72:                                     end for
73:                                     end if
74:                                     if id1-2 ≥ 0 and left = -1 and «,» ∉ l[id1-2 : id1] then
75:                                     for from i = 0 to len(resl2)-1 do
76:                                     r ← check(l[id1-2 : id1])
77:                                     if «N» ∈ r then
78:                                     return [«он», «писали» ]
79:                                     else if «Y» ∈ r then
80:                                     left ← id1-3
81:                                     break
82:                                     end if
83:                                     end for
84:                                     end if
85:                                     end if
86:                                     end if
87:                                     end for
88:                                     end if
89:                                     end if
90:                                     end if
91: end function

```

---



## Список литературы

- [1] Журавлёв, Ю.И. Дискретный анализ. Формальные системы и алгоритмы: Учебное пособие / Ю.И. Журавлёв, Ю.А. Флёров, Н.М. Вялый – М.: ООО Контакт Плюс, 2010. – 336 с.: ил.
- [2] LanguageTool — Проверка грамматики и стилистики [Электронный ресурс] — <https://languagetool.org/ru>

## ПРИЛОЖЕНИЕ А

### Система правил для анализа словосочетаний из двух слов

Таблица 1 – Система правил для словосочетаний из двух слов

№	prt_l	sing_l	cow_r	prt_r	sing_r	cow_l	ans	example
1	b	N	1	5	N	–	Y	мы делали
2	1	Y	1	5	N	–	N	собака лаяли
3	1	Y	1	5	Y	–	Y	самолёт летит
4	b	Y	1	5	Y	–	Y	я делаю
5	6	–	–	1	Y	4	Y	делать дело
6	5	Y	–	6	–	–	Y	хочет есть
7	6	–	–	b	Y	2	Y	знать его