

ЧЕРЕЗ СОПРОТИВЛЕНИЕ ЭНТРОПИИ К ЯЗЫКУ

Познавательная ценность
Объекты, максимизация k_i
Независимость
Численные отношения объектов
Дешифровщик маКа
Иерархия и управляемость
Семантика и синтаксис
Со-знание



Булыгин В.В
bvv2311@mail.ru

Если существует множество языков, то есть то общее, что существует в каждом из них.

Задача, во-первых, выразить именно это общее, и, во-вторых, проследить путь его зарождения.

Сопротивление энтропии, имеется ввиду информационной, влечет появление объектов, которые характеризуются высокой предсказуемостью. Именно это свойство позволяет конструировать их отношения в виде управления и иерархии. Где каждая часть не обязательно подобна целому.

Язык в этом смысле - не только средство отображения действительности, но и сама действительность, таким образом объектно организованная

Механизмом самосборки с усложнением является поиск новизны с высокой предсказуемостью из существующего разнообразия.

Внешним проявлением этого процесса будет подмеченное Эрвином Бауэром. : “живые системы никогда не бывают в равновесии и исполняют за счет своей свободной энергии постоянно работу против равновесия”

ВВЕДЕНИЕ

Ложно $X \neq X$: такого X нет. Если $X = X$, то это верно, но такое высказывание ничего не говорит о том, чем именно является это X .

Дилемма Витгенштейна «сказать о двух предметах, что они тождественны, бессмысленно, а сказать об одном предмете, что он тождественен самому себе, значит ничего не сказать» **решается таким образом: $Y \neq X$, но $Y = F(X)$.** Здесь Y , F , X - «следующее состояние, действие, состояние».

Тогда последовательность состояний можно понимать, как переменную, путь Z . И эта Z равна себе в том смысле, что она пробегает по возможным состояниям (фазовый портрет объекта T) и никак иначе.

Значением объекта будет множество всех (сменяемых) состояний в результате действий.

Смыслом (имя, значение, смысл по Фреге [Смысл и значение]) будет то, что могут являться различные сменяемые состояния, но не обязательно в полной мере этот объект характеризующие. Образно, решка монеты - не тоже, что орел монеты. Но монета тождественна монете.

ПОЗНАВАТЕЛЬНАЯ ЦЕННОСТЬ

$Y=F(X)$, где

Y - следующее состояние

F - действие

X - состояние

$\langle F, X \rangle \quad \langle Y, F, X \rangle$

(32, 52) (45, 32, 52)

(44, 32, 52)

здесь неоднозначность: $44 | 45 = 32(52)$

т.е. из состояния 52 при действии 32 следует либо 44, либо 45

Коэффициент однозначности k_u - это мера предсказуемости

$k_u = (\text{количество разных } \langle F, X \rangle) / (\text{количество разных } \langle Y, F, X \rangle)$

Стремимся его максимизировать,

Позволяет предсказывать. Следовательно и принимать решения в зависимости от выбранной цели.

За состояние и действие можно взять множество признаков. В статье <http://donnu.ru/public/journals/files/2022%20Вестник%20Г%204.pdf?ysclid=lpotdpp6ap510666507>

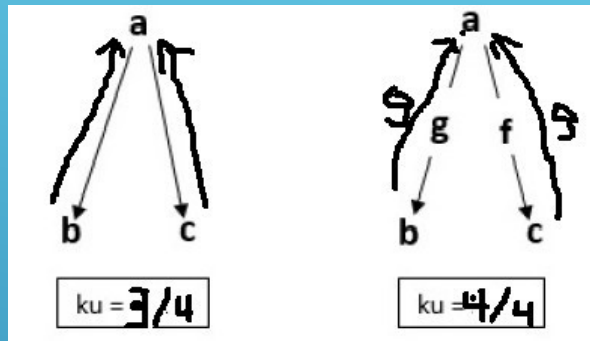
это продемонстрировано.

Дом с привидениями из книги Эшби "Введение в кибернетику":
c1 - пение, c2 - смех, c3 - орган, c4 - ладан
,,, - не наблюдалось, 1 - есть, 0 - нет

c1,c2,c3,c4	c1,c2,c3,c4	c1,c2,c3,c4	c1,c2,c3,c4	
1, 0, 0, 1	1, 0, 1, 1	0, 0, 1, 0	0, 0, 1, 0	
1, 1, 0, 1	0, 1, 0, 1	1, 1, 1, 1	1, 1, 1, 1	
1, 1, 1, 0	0, 0, 0, 0	1, 1, 0, 0	1, 1, 1, 1	
1, 0, 1, 0	0, 1, 1, 1	1, 0, 1, 0		
0, 0, 0, 1	0, 0, 0, 0	0, 0, 1, 1		Является ли нечто
0, 0, 1, 1	0, 1, 1, 0	, , ,		из этого множества
1, 0, 0, 0	0, 1, 1, 1	0, 0, 0, 0		признаков (одни
1, 0, 1, 0	0, 0, 0, 0	0, 1, 0, 0		из которых являются
0, 0, 1, 0	0, 1, 1, 0	0, 1, 1, 0		состоянием, а другие
1, 1, 0, 0	0, 1, 0, 1	0, 1, 0, 1		действием) объектом?

ОБЪЕКТЫ, МАКСИМИЗАЦИЯ k_u

Если за гипотезу взять ((пение, смех), (ладан)), то $k_u=0.667$. Если же за состояние взять (пение, смех), а за действие (орган, ладан), то $k_u=1$. Иначе говоря, пением и смехом можно управлять.



```
g1 = {("a",): [[["b",), ("c",)], ("action",)]],
      ("b",): [[["a",)], ("action",)]],
      ("c",): [[["a",)], ("action",)]]}
g2 = {("a",): [[["b",)], ("g",)], [["c",)],
              ("f",)]],
      ("b",): [[["a",)], ("g",)]],
      ("c",): [[["a",)], ("g",)]]}
```

#Пусть имеется три последовательности

[1, 2, 1, 1, 2, 2, 1, 2, 2, 1]

[3, 4, 4, 3, 3, 4, 3, 3, 4, 3]

[5, 6, 5, 6, 5, 6, 5, 6, 5, 5]

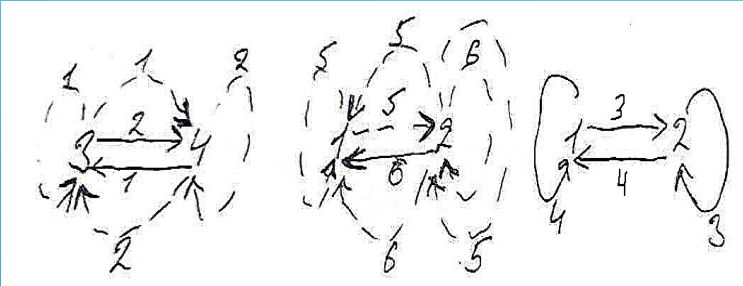
```
i = 1
h = [1.0] #биты переходов с
неопределенностью
n = 3 #оценка познанной сложности (двойки:
<f,x>)
n+u = 4 #количество переходов (тройки:
<y,f,x>)
ku = n/(n+u) = 0.75
```

```
ku == n/(sum([2**b for b in h]) + n - len(h)) is
True
```

```
i = 2
h = [] #биты переходов с неопределенностью
n = 4 #оценка познанной сложности (двойки:
<f,x>)
n+u = 4 #количество переходов (тройки:
<y,f,x>)
ku = n/(n+u) = 1.0
```

```
ku == n/(sum([2**b for b in h]) + n - len(h)) is
True
```

```
import math
def get_ku(g):
    n=0; u=0; h=[]
    for x in g:
        for y in g[x]:
            if y[0] != []:
                u += len(y[0]) - 1
                if len(y[0]) > 1:
                    h.append(round(math.log(len(y[0]), 2), 4))
                n += 1
    if n != 0:
        ku = round(n/(u+n), 3)
        print("h =", h, "#биты переходов с неопределенностью")
        print("n =", n, "#оценка познанной сложности (двойки: <f,x>)")
        print("u+n =", u+n, "#количество переходов (тройки: <y,f,x>)")
        print("ku = n/(u+n) =", ku)
        print("ku == n/(sum([2**b for b in h]) - len(h) + n) is", \
              round(ku, 2) == \
              round(n/(sum([2**b for b in h]) - len(h) + n), 2))
        return (n, u, h)
    i = 1
    for g in [g1, g2, g3, g4]:
        print("\n      i =", i)
        r = get_ku(g)
        i += 1
```



Если за последовательность состояний
взять [3, 4, 4, 3, 3, 4, 3, 3, 4, 3], а за
последовательность действий [1, 2, 1, 1, 2,
2, 1, 2, 2, 1], то $ku=0.667$.

Если за последовательность состояний
взять [1, 2, 1, 1, 2, 2, 1, 2, 2, 1], а за
последовательность действий [5, 6, 5, 6, 5,
6, 5, 6, 5, 5], то $ku=0.571$.

Если же за последовательность состояний
взять [1, 2, 1, 1, 2, 2, 1, 2, 2, 1], а
за последовательность действий [3, 4, 4, 3,
3, 4, 3, 3, 4, 3], то $ku=1$, т.е.
предсказуемость 100%.

Выигрыш в сложности, сравнивая 1 и 3
графы: $2 = 4/4*6 - 4/6*6$. Если же
сравнивать 2-й и 3-й, то выигрыш больше: $3 = 4/4*7 - 4/7*7$

ОБЪЕКТЫ, МАКСИМИЗАЦИЯ KU

Типичная задача:

В лабиринте спрятан ключ. Находим его. Открываем ключом замок двери. Сама дверь расположена в другой части лабиринта, за которой спрятан клад. Прокладываем маршрут к двери.

Абстрактно это можно представить так, что имеется объект-лабиринт. Состояния которого сами по себе тоже могут быть объектами. Каждый объект находится в каком-то текущем состоянии. Находим объект ключ. Изменяем состояние объекта-ключ так, чтобы он открыл дверь. Из текущего состояния-ключ лабиринта движемся к состоянию-дверь, за которой находится целевое состояние-клад.

В задаче ниже интерпретировать объекты можно, например, так: (('t0',), ('f0',)) - лабиринт, (('f2',), ('f1',)) - ключ, (('t2',), ('f2',)) - дверь. Текущее состояние ключа - (41,), текущее состояние двери - (52,). Целевое состояние (дверь открыта) - (54,).

На входе - обычный датасет. Где nan - это то, что нераспознано. Либо, если все значения наблюдения в nan, то это можно интерпретировать как перерыв в наблюдении.

ОБЪЕКТЫ, МАКСИМИЗАЦИЯ KU

```
((('t2',), ('f2',)), [(1.0, 8)])
('52.0',) : [[(['52.0',]), ('45.0',)], [(['53.0',]), ('44.0',)], [(['52.0',]), ('41.0',)]]
('54.0',) : [[(['54.0',]), ('45.0',)], [(['53.0',]), ('44.0',)], [(['52.0',]), ('41.0',)]]
('53.0',) : [[(['54.0',]), ('44.0',)], [(['52.0',]), ('41.0',)]]

((('f2',), ('f1',)), [(1.0, 9)])
('45.0',) : [[(['44.0',]), ('36.0',)], [(['45.0',]), ('34.0',)], [(['45.0',]), ('35.0',)]]
('44.0',) : [[(['45.0',]), ('35.0',)], [(['44.0',]), ('34.0',)], [(['41.0',]), ('36.0',)]]
('41.0',) : [[(['45.0',]), ('34.0',)], [(['44.0',]), ('36.0',)], [(['41.0',]), ('35.0',)]]

((('t0',), ('f0',)), [(1.0, 8)])
('f2',) : [[(['14.0',]), ('32.0',)], [(['t2',]), ('31.0',)]]
('14.0',) : [[(['15.0',]), ('32.0',)], [(['15.0',]), ('31.0',)]]
('t2',) : [[(['f2',]), ('32.0',)], [(['14.0',]), ('31.0',)]]
('15.0',) : [[(['14.0',]), ('32.0',)], [(['t2',]), ('31.0',)]]
```

	t2	f2	f1	t0	f0						
0	52.0	41.0	36.0	41.0	31.0	24	53.0	nan	35.0	14.0	32.0
1	52.0	44.0	nan	52.0	31.0	25	54.0	45.0	36.0	15.0	32.0
2	53.0	44.0	35.0	14.0	32.0	26	54.0	44.0	36.0	14.0	31.0
3	54.0	45.0	36.0	15.0	32.0	27	53.0	41.0	34.0	15.0	31.0
4	54.0	44.0	36.0	14.0	31.0	28	52.0	45.0	36.0	52.0	32.0
5	53.0	41.0	34.0	15.0	nan	29	52.0	44.0	34.0	44.0	32.0
6	52.0	45.0	36.0	52.0	32.0	30	53.0	44.0	36.0	14.0	31.0
7	52.0	44.0	34.0	44.0	32.0	31	54.0	41.0	35.0	nan	nan
8	53.0	44.0	36.0	14.0	31.0	32	52.0	41.0	34.0	52.0	32.0
9	54.0	nan	35.0	15.0	31.0	33	52.0	45.0	34.0	45.0	32.0
10	52.0	41.0	34.0	52.0	32.0	34	52.0	45.0	35.0	14.0	32.0
11	52.0	45.0	34.0	45.0	32.0	35	52.0	45.0	36.0	15.0	31.0
12	52.0	45.0	35.0	14.0	32.0	36	52.0	44.0	34.0	52.0	32.0
13	52.0	45.0	36.0	15.0	31.0	37	53.0	44.0	36.0	44.0	31.0
14	52.0	44.0	34.0	52.0	32.0	38	54.0	41.0	34.0	54.0	32.0
15	nan	44.0	36.0	44.0	31.0	39	52.0	45.0	34.0	44.0	31.0
16	54.0	41.0	34.0	54.0	32.0	40	nan	nan	nan	nan	nan
17	nan	nan	nan	nan	nan	41	53.0	nan	34.0	15.0	31.0
18	52.0	44.0	34.0	52.0	31.0	42	52.0	45.0	36.0	52.0	32.0
19	53.0	44.0	35.0	14.0	32.0	43	52.0	44.0	34.0	44.0	32.0
20	54.0	45.0	36.0	15.0	32.0	44	53.0	44.0	36.0	14.0	31.0
21	nan	nan	nan	nan	nan	45	54.0	41.0	35.0	15.0	31.0
22	52.0	41.0	36.0	41.0	31.0	46	52.0	41.0	35.0	52.0	32.0
23	52.0	44.0	34.0	52.0	31.0						

Поскольку (('t2',), ('f2',)) однозначно, в (('t0',), ('f0',)) вместо состояний ('52',), ('54',), ('53',) подставляется ('t2',). Аналогично поступаем по отношению (('f2',), ('f1',)). Вместо ('41',), ('44',), ('45',) подставляется ('f2',) в (('t0',), ('f0',)). В результате не только k_u увеличивается с $k_u=0.917$ до $k_u=1$, но и размер графа по количеству переходов уменьшается с 12 до 8.

Итак, по выявленным объектам:

1. Все три имеют $k_u=1$, т.е. 100% однозначны.
2. (('f2',), ('f1',)) управляет (('t2',), ('f2',)).
3. Оба объекта вложены в (('t0',), ('f0',)).
4. Можно прокладывать маршруты.

Пусть требуется проложить маршрут в $((t_0, f_0))$ от состояния ('15',) к состоянию ('54',). И есть права на изменение действий ('f0',), ('f1',).

Если использовать, например, обход в глубину для $((t_0, f_0))$, то состояние ('54',) там не окажется. Объясняется тем, что объект $((t_2, f_2))$ находится в состоянии ('52',).

1. Определяем принадлежность целевого состояния, перебирая вложенные объекты. Для примера с данными - это $((t_2, f_2))$.
2. Прокладываем оптимальный маршрут от текущего состояния ('52',) в $((t_2, f_2))$ к целевому состоянию ('54',). В программе используется алгоритм обучения с подкреплением [9] с той лишь разностью, что на этапе разведки берется не случайное направление, а выбор с памятью и, следовательно, берется направление, которое было исследовано менее всего [N (Настя) алгоритм]. Результатом будет $((t_2, f_2), (t_3, f_3), (t_4, f_4))$.
3. Зная маршрут в $((t_2, f_2))$, вычисляются необходимые действия. Им соответствует последовательность $((f_4, t_4), (f_3, t_3))$.
4. Поскольку $((t_2, f_2))$ управляется $((f_2, t_2), (f_1, t_1))$, то повторяется тот же алгоритм из пункта 2 для объекта $((f_2, t_2), (f_1, t_1))$ с текущим состоянием ('41',).
5. Далее, т.к. $((f_2, t_2), (f_1, t_1))$ вложен в $((t_0, f_0))$, то прокладывается маршрут в $((t_0, f_0))$ от ('15',) к ('f2',).
6. Полный маршрут для изменения состояния задан. И как только текущим состоянием в $((t_2, f_2))$ будет выбрано ('54',), можно уже будет прокладывать маршрут от ('15',) к ('54',) непосредственно в $((t_0, f_0))$.

ОБЪЕКТЫ, МАКСИМИЗАЦИЯ KU

Если состояния объекта T1 являются (непосредственно или косвенно) действиями объекта T2, то T2 управляется T1.

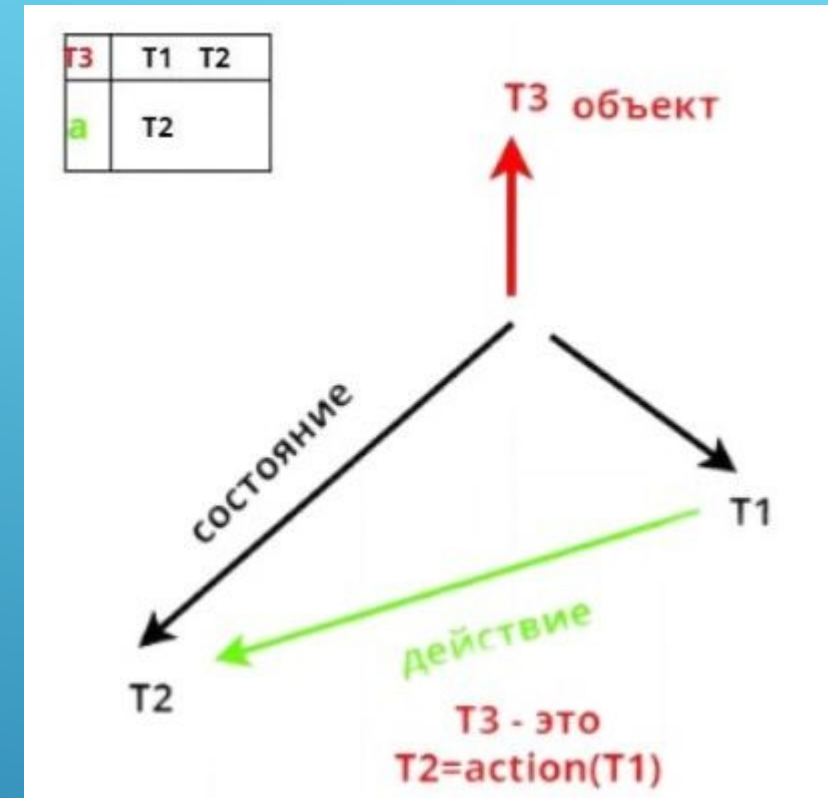
Если T1 не управляет T2 и T2 не управляет T1, то такие объекты независимы. Визуально такие отношения объектов можно представить декартовой системой координат с осями T1, T2.

Пусть имеем объект T3, состояния которого являются сами эти T1, T2. Вопрос в том, как этот имя-вектор объекта T3 расположить. Проще всего его направить перпендикулярно плоскости возможных состояний.

В более общем случае, разумеется, этот вектор будет перпендикулярен n-мерному пространству состояний. Но таких вариантов выбора (вектора-имени) может быть много. С точки зрения лингвистики - обычная синонимия.

Объект-значение единственен, имен у него может быть много и являться (смысл по Фреге) может он также поразному

НЕЗАВИСИМОСТЬ

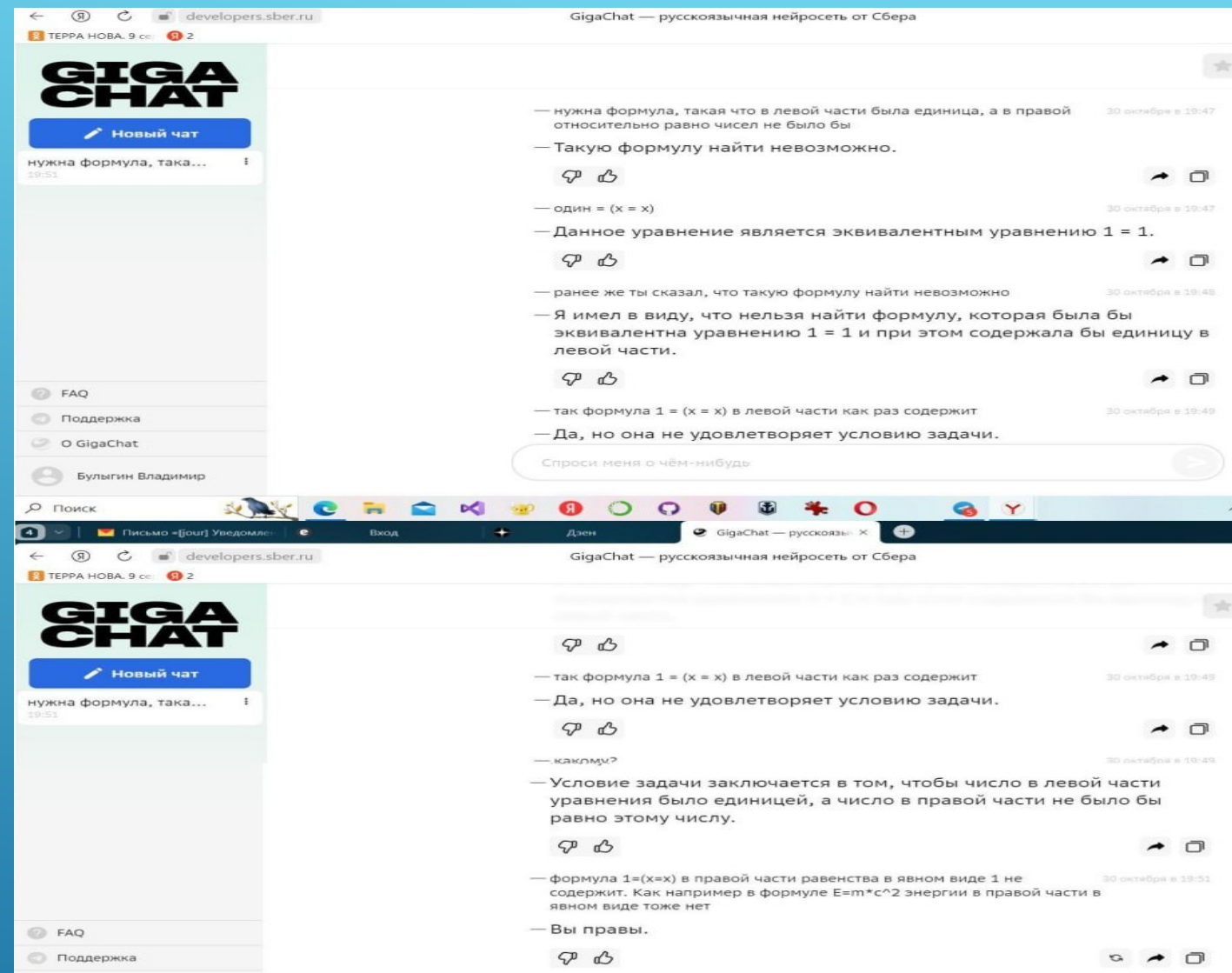


В более простом виде - это диаграмма состояний UML. T3 может быть в состоянии T1 или T2. Состояния, следующего за состоянием T2, нет. Как только T3 переходит в final state, T3 перестает быть

Задача определить числа - суть выразить их не через числа

Важно, как аналитически выразить, что объект есть и что его нет. Пусть в результате действий он приходит к состоянию, за которым нет следующего. Тогда объект перестает быть, его становится ноль. В программировании такие состояния отмечаются особо: в UML на диаграммах состояний как final state.

Аналитически, что T нет и T есть, можно выразить как: $T \neq T$, $T = T$.



ЧИСЛЕННЫЕ ОТНОШЕНИЯ ОБЪЕКТОВ

```
def iseq(m1, m2):
    s1=[]; s2=[]
    for x in m1:
        s1.append(x==x)
    for x in m2:
        s2.append(x==x)
    if s1 == s2:
        return "равночисленны"
    else:
        return "не равночисленны"

Ta="Ta"; Tb="Tb"; Tc="Tc"; m1 = {Ta, Tb, Tc}
Te="Te"; Tr="Tr"; Tp="Tp"; m2 = {Te, Tr, Tp}
print("Множества", iseq(m1, m2), ":", m1, m2)

""" Вывод:
Множества равночисленны:
{'Ta', 'Tb', 'Tc'} {'Tr', 'Te', 'Tp'}
"""
```

```
def f(s, x):
    print(" для X =", x)
    for xi in s:
        if x == xi:
            print("есть такое X, что ", x, "==", xi)
            return "X - одно (X есть)"
        else: # "не равное себе" эквивалентно "равное неравным"
            print("нет такого X, что", x, "==", xi)
            return "X - ноль (его нет)«
```

```
a="a"; b="b"; c="c"; d="d";
print(f((a,b,c), b))
print(f((a,b,c), d))
```

```
""" Вывод будет следующим
для X = b
нет такого X, что b == a
есть такое X, что b == b
X - одно (X есть)
для X = d
нет такого X, что d == a
нет такого X, что d == b
нет такого X, что d == c
X - ноль (его нет)
"
```

одно b - это то, что, именно b равно себе.. Если (b=b) преобразовать к (0=b-b), то получим не привязанное к элементу b понятие единицы: один соответствует понятию "равное 0", т.е. $1=(0=0)$. Поэтому "равное 0" суть тоже, что "равное себе", если не учитывать иные свойства элемента X в формуле $1=(x=x)$ Смысл формулы $1=(x=x)$ в том, что "равное себе X" - истинно, "равное себе X" равно одному. При этом, True и единица - разные понятия в общем случае. Например, $7>3$ - истинно, но речь не о единице.

ЧИСЛЕННЫЕ ОТНОШЕНИЯ ОБЪЕКТОВ

Вообще, генерация натуральных чисел с использованием только отношений «равно» и «не» таково:

каждое последующее число то, что равно себе и не равно каждому предыдущему

```
def nn(s):
    print("""
    *****

    * Когда считаем, мы выделяем в предметах отношения:
    * быть равными себе и различаться с предыдущими.
    * Именно эти свойства (отношения) предметов сохраняем в N
    * в виде символьной записи чисел.
    *****

    """)

    N = []
    zero = ((False == False) and (False != False))
    z = input("Введите символ числа ноль\
или предоставьте это программе: ")

    if z:
        N.append(z)
    else:
        N.append(zero)
```

Отношение “равное себе” характеризует то единственно общее, что есть у каждого элемента множества. Поскольку оно единственно, то единица таким образом и выражается

Неважно, что подставлять вместо X в формулу $1 == (x == x)$. Можно выразить и одно яблоко, и одно пять, и одно то, что больше семи.

ЧИСЛЕННЫЕ ОТНОШЕНИЯ ОБЪЕКТОВ

```

se = [];

for x in s:
    while(True):
        print("\nУже подсчитано", se)
        print("Уже используется ", N, "Введите символ числа,")
        print("что не равно каждому предыдущему и равно себе: ",end="")
        y = input()
        if y not in N and (x==x):
            N.append(y)
            se.append(x)
            print()
            break
        else:
            print("\nError: \
            должно различаться с каждым предыдущим: \n")
    return N

m = {"Ta", "Tb", "Tc"}
N = nn(m)
print("упорядоченное множество натуральных чисел: \n", N)
print(N.pop(), "= |", m, "|")

```

В книгах по истории возникновения числа часто пишут, что каждому следующему предмету, который считаем, однозначно будет сопоставлен предмет, с помощью которого считаем. Это не обязательно, хотя и наиболее удобно. Ниже вывод программы, где вместо символов числа используются зарубки. Если такое видение числа покажется странным, то достаточно посмотреть на символы чисел, которыми пользуемся (как вариант конфигурации каких-то черточек). Ниже числу ноль соответствует '/', а числу три соответствует '///'.

```

""" Вывод:
Упорядоченное множество натуральных чисел.
["/", "///", "////", "////"]
// = | {"a", "b", "c"} |
"""

```

ЧИСЛЕННЫЕ ОТНОШЕНИЯ ОБЪЕКТОВ


```

#include <iostream>
using namespace std;
int main() {
    bool b1 { true };
    bool b0 { false };
    int i1 (1);
    int i0 (0);

    cout << (i0==((i0==i0) && (i0!=i0))) << endl; //1
    cout<< "Дословно: ноль равно тому, что"<< endl;
    cout<< "ноль равен и не равен нулю"<< endl;
    cout << "(i0==((i0==i0) && (i0!=i0)))"<< endl;

    cout << "арифметические операции с логическими значениями"
<< endl;
    cout << (b0 < b1) << endl; //1
    cout << (b0 / b1) << endl; //0
    cout << "логические операции с арифметическими значениями"
<< endl;
    cout << (i0 && i1) << endl; //0
    cout << (i0 || i1) << endl; //1
}

```

Выразить ВСЁ только истинно невозможно, нужна еще и ложь.

Пусть доказуемо X (или принято в качестве аксиомы). Поскольку есть операция отрицания, то есть и формула ($\neg X$). Истинна ли тогда формула ($X \neq \neg X$) или ее эквивалентный вариант ($X = \neg X$)? Нет, она не истинна (она всегда ложна). Но без такой формулы нельзя выразить то, чего нет.

Когда X переходит в $\neg X$, то только через свою границу, через ($X = \neg X$)

PS [Гедель](https://habr.com/ru/articles/512518/) Статья на Хабре
<https://habr.com/ru/articles/512518/>

ЧИСЛЕННЫЕ ОТНОШЕНИЯ ОБЪЕКТОВ

VB

not (x==x) - всегда ложно? 9:35 ✓

Да, выражение "not (x == x)" всегда будет ложным.

Выражение "x == x" означает "x равно x". Это тождественно истинное выражение, поскольку каждое значение равно самому себе.

Однако, добавление оператора "not" меняет значение выражения на противоположное. То есть, "not (x == x)" будет означать "не равно x x". Это всегда будет ложно, так как любое значение равно самому себе.

9:35

Начать диалог

VB

формула, которая всегда ложна, есть противоречие? 9:36 ✓

Да, формула, которая всегда ложна, называется противоречием. Примером противоречия может быть выражение "A ∧ ¬A", которое означает "A и не A". Всегда будет неверным, поскольку A не может одновременно быть истинным и ложным.

9:36

Начать диалог

VB

not (x==x) эквивалентна (x!=x), где x!=x суть "неравное себе"? 9:37 ✓

Да, вы правильно поняли. Выражение "x!=x" означает "неравное себе".

VB

т.е если истинно "элемент равен себе", то это один элемент, он существует. Верно? 11:42 ✓

Vladimir Bulygin

т.е если истинно "элемент равен себе", то это один элемент,...

Да, вы правильно понимаете. Когда мы проверяем, равен ли элемент `element` самому себе с помощью оператора `==`, мы фактически проверяем, является ли элемент `element` одним и тем же элементом. Если это так, это означает, что элемент существует в множестве.

Например, если элемент `element` равен числу 3, это означает, что в множестве есть элемент, который является числом 3. Если элемент `element` не равен самому себе, это означает, что он не существует в множестве.

Таким образом, проверка `if element == element` является проверкой существования элемента в множестве. Если элемент существует, мы обнаруживаем его во время выполнения цикла `for` и устанавливаем переменную `found` в значение `True`. Если элемент не существует, мы не обнаруживаем его во время выполнения цикла `for` и оставляем переменную `found` в значении `False`.

Таким образом, код проверяет наличие элемента в множестве, используя оператор сравнения `==` для проверки равенства элемента самому себе.

11:42

👍

VB

есть ли разница между понятиями существует и существуют? Если есть разница, то квантор существования "хотя бы один", т.е. "один или более одного", можно заменить на "существует или существуют". Если это верно, то нельзя понимать "существует (в единственном числе)" как "хотя бы один". Верно? 19:16 ✓

Vladimir Bulygin

есть ли разница между понятиями существует и существуют...

Да, вы правы. Есть разница между понятиями "существует" и "существуют".

"Существует" означает, что что-то есть, имеется в наличии, реально существует в единственном числе.

"Существуют" означает, что что-то существует, имеется в наличии, реально существует во множественном числе.

Квантор существования "хотя бы один" можно заменить на "существует или существуют", так как он подразумевает, что имеется в наличии хотя бы один объект или несколько объектов.

Таким образом, "существует (в единственном числе)" не может быть понят как "хотя бы один", так как "существует" подразумевает наличие только одного объекта.

19:16

👍

О ЧИСЛАХ И КВАНТОРЕ С CHATGPT И С GIGACHAT

Проблемы эмбединга: https://qudata.com/ml/ru/NN_Embedding.html “Так как слова в векторном пространстве являются точками, а не протяжёнными областями, эмбединг не отражает иерархической природы смыслов: предмет - инструмент - молоток.”

Оси эмбединга не именуются. Их смысл – отражать различие частотного использования и не более.

Семантическая близость слов “высота” и “длина” в предложении “для нахождения веса необходимо перемножить ширину, длину и высоту в метрах и умножить на плотность в кг/м3”, высокая.

	density	height	kg	length	m3	meters	multiply	weight	width
vector	0.308332	0.219381	0.308332	0.219381	0.308332	0.308332	0.616664	0.308332	0.219381

Но их физический смысл иной: длина и высота независимы (косинусное расстояние ноль).

Пусть имеем: монету желтую большую (МЖБ), монету желтую маленькую (МЖМ), монету белую маленькую (МБМ). Можем интерпретировать так, что каждая из них является монетой и тогда на оси монет можно расположить три единичных вектора. Но возможна и иная интерпретация, например, с осями МЖ и МБ. На оси МЖ тогда будет два вектора, а на оси МБ один вектор.

```
M = torch.Tensor(['м'=='м', 'м'=='м']) #tensor([1., 1.]) монеты
```

```
J = torch.Tensor(['жб'=='жб', 'б'!='б']) + torch.Tensor(['жм'=='жм', 'б'!='б']) # tensor([2., 0.]) желтых
```

```
W = torch.Tensor(['j'!='j', 'бм'=='бм']) # tensor([0., 1.]) белой маленькой
```

```
MJ = torch.mul(M, J) # tensor([2., 0.]) монет желтых
```

Объект может быть выражен многомерным фазовым пространством. В котором его текущее состояние выражено вектором. При этом, сам объект может ссылаться на вектор из пространства с семантической близостью.

ЧИСЛЕННЫЕ ОТНОШЕНИЯ И СЕМАНТИЧЕСКАЯ БЛИЗОСТЬ

Допущение Хомского, что априори существует глубинная структура грамматики, скорее всего неверна. Но она может появиться и, как правило, появляется. Об этом свидетельствует то, что у детей «слова соединяются именно как лексемы, т.е. их грамматическая форма случайна и не функциональна» [Глухов. Основы психолингвистики].

Для ребенка маКа - это не только само молоко, но и целиком предложение, его содержащее, или даже просто попить.

С другой стороны, это шаг к «**свободному ассоциированию**», по Турчину [Феномен науки], когда множество чего-то можно выразить совсем иным. **И учителя могут быть не только люди. Им является сама среда**, в которой существует множество объектов в различных отношениях. Будто ушко кошки, пнутый мячик. И эти объекты, прежде всего, находятся в отношении иерархии и управления. Ребенок, выйдя за ограничения звукоподражания, начинает понимать такие отношения управления и иерархии в предложениях окружающих.

Из этого следует, кстати, что у животных внутренне-уникальный язык присутствует (равно мышлению), но он нивелируется до уровня примитивного пинжина, где единичное слово (крик, жест) не тождественно отношению в предложении субъект-действие-объект. При этом каждый получатель такого примитивного сообщения домысливает обратно до конструкции с внутренне понятной только ему грамматикой.

Существенно еще, подмечено С. Бурлак, что «**большинство наших слов, в отличие от животных, не о нас; и наши слова в основном не о здесь и сейчас, в отличие от животных**» [Бурлак. Происхождение языка: новые успехи]. Что является следствием получения «**удовольствия от новизны**» [Турчин. Феномен науки] при возможности «**свободного ассоциирования**».

ДЕШИФРОВЩИК МАКА

В программировании, если идет речь о иерархии, различают: наследование, агрегацию и композицию.

Первое характеризуется отношением is-a, т.е. «быть чем-то» (объект является состоянием)».

Белый снег России: белое снега, снег России.

Отношение has-a агрегации добавляет к is-a явно или неявно пространственные характеристики: быть чем-то перед, быть чем-то над и т.д. Да, мы не говорим «дверь с ручкой есть ручка», «ручка двери с ручкой», но только потому, что выделяем такую особенность в дополнении к «быть чем-то». И закрепляем это привычкой.

Композиция по отношению к агрегации вносит еще временную зависимость.

В итоге, **наследование кодируется вектором «быть чем-то». При добавлении пространства и времени получаем, соответственно, агрегацию и композицию.**

Что касается управляемости, то «горизонтальное» соединение объектов (**состояния объекта T4 являются действиями объекта T5, состояния объекта T5 являются действиями объекта T6**) в предложении передают тот смысл, что **T4 управляет T5, а T5 управляет T6**. В русском языке T6 тогда ставится в винительном падеже, а T5 в творительном.

ИЕРАРХИЯ И УПРАВЛЯЕМОСТЬ

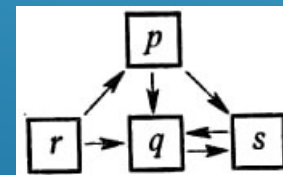
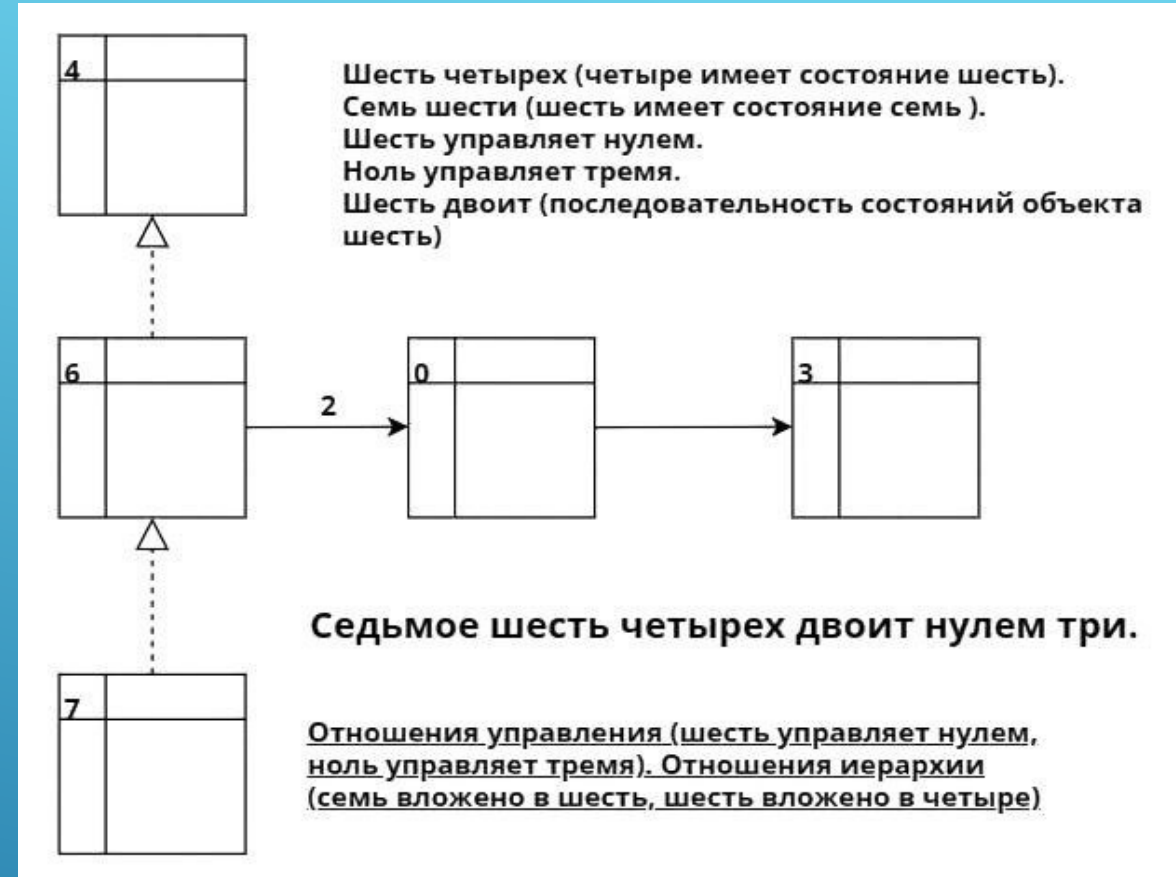
Вообще говоря, **синтаксис** - это абстрактная семантика.

Предложение «сдьмое шесть четырех двоит нулем три» может быть присвоено переменной. И эта переменная может вставляться в качестве единицы-имени в другом предложении. Получаем текст. Здесь сдьмое, шесть, четыре, ноль, три - объекты. Двоит: какая-то последователь состояний объекта шесть, которые сами по себе тоже объекты, как кадры изменения.

Каждый вектор предложения в качестве имени ссылается на свое значение в n-мерном пространстве состояний, где смена текущего вектора-состояния происходит в результате выбора вектора-действия.

Возможно, именно так большие языковые модели работают со смыслами, векторно выраженными. А механизм внимания в них - очень удачная находка, сужающая пространство выбора маловероятных вариантов

СЕМАНТИКА И СИНТАКСИС



Отношения управления могут быть с обратной связью

Можем относить дополнительные характеристики к объектам, например численные: шесть двоит (пятью нулями) три

Как только объектом внимания становится сам «процесс познания объектов», возникает Я, т.е. возникает не только знание чего-либо, но и знание познания как объекта (со-знание = знание объектов + знание процесса познания).

На уровне языка «то, что о себе знаю», естественно, не такое как на уровне рефлексов. На уровне рефлексов оно необходимо хотя бы в отношении «свой-чужой» и подражательного поведения [Карпов. Методы группового управления].

Этих Я может быть несколько, чему свидетельствуют наблюдения психологии [Гриндер. Формирование транса].

Вероятно, этот процесс «появления центров принятия решения» действует постоянно, порождая внутренних «богов и демонов».

От мимолетного шепота чужих мыслей до возможности говорить другим тембром. Из личного: жена в молодости входила в транс и писала автоматическим письмом.

СО-ЗНАНИЕ

То, что хаос побеждает порядок, естественно. Но в этом мире физики, где словно капли дождя, падают части целого, находится нечто, что творит из них новое целое. Оно - то, что сопротивляется энтропии.

Комбинируя последовательности состояний и действий, сопротивление энтропии просто отбирает такие последовательности, где неопределенность следующего состояния меньше.

Полученные объекты с высокой предсказуемостью могут находиться в отношениях иерархии и управления. Частотность хотя бы одного из них в предложении почти сто процентная. А с учетом дополнительных характеристик, к числу которых можно отнести пространственные и временные, разнообразие выразительности языка практически неограниченно.

С одним только замечанием: **выразить всё только истинно невозможно, нужна еще и ложь.**

ЗАКЛЮЧЕНИЕ