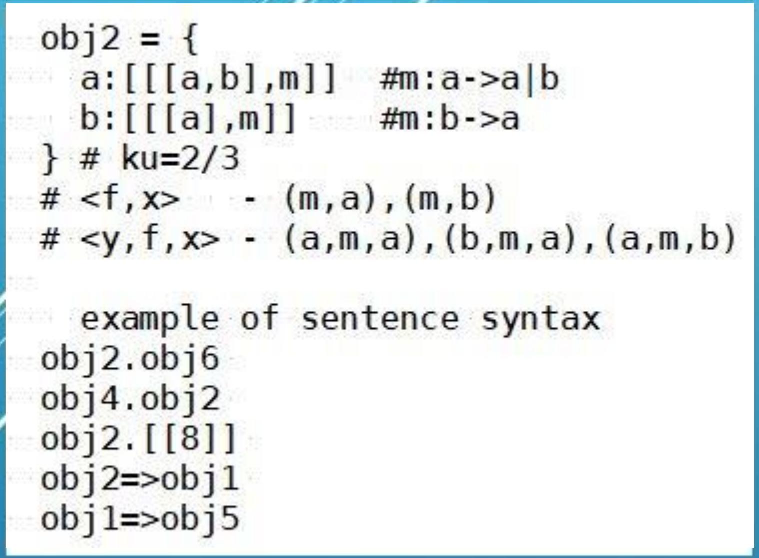
CYBERNETIC APPROACH TO UNDERSTANDING LANGUAGE

ABSTRACT

The purpose of the article is to express what language is and how it arises. An approach from the point of view of cybernetics. It is demonstrated that the search for objects can be carried out by means of reinforcement learning.

Such objects, maximized by the coefficient of uniqueness, can enter into relationships. Combinations of basic management relationships and hierarchies make it possible to build proposals. Which allows you to model reality and serve as a means of communication.



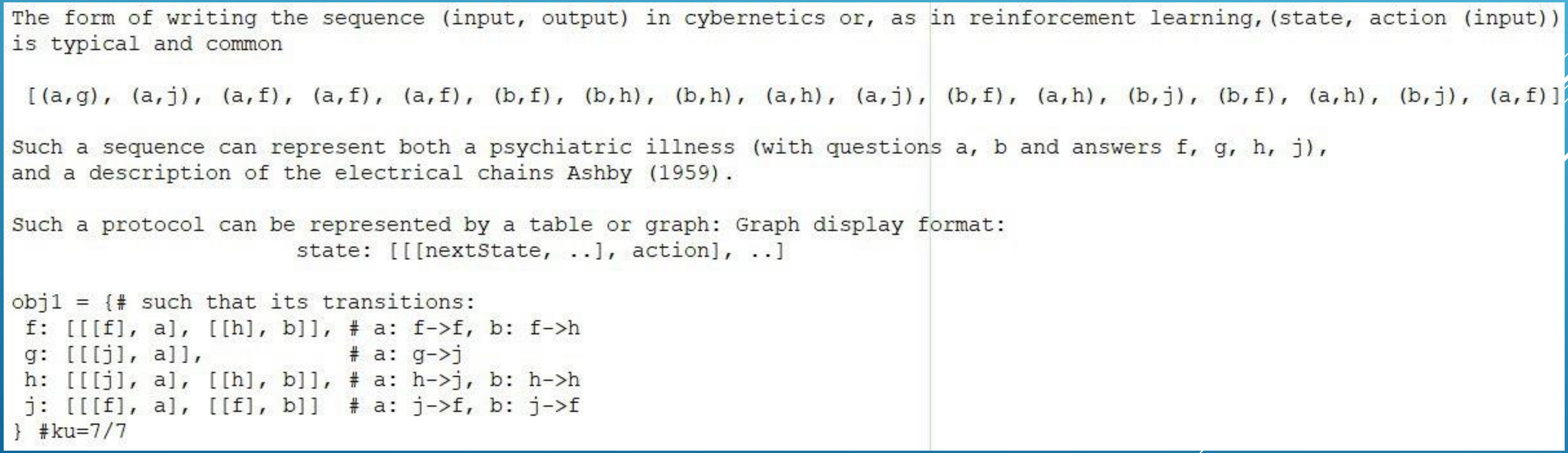
```
obj2 = {
  a:[[[a,b],m]]  #m:a->a|b
  b:[[[a],m]]    #m:b->a
} # ku=2/3
# <f,x>   - (m,a),(m,b)
# <y,f,x> - (a,m,a),(b,m,a),(a,m,b)

  example of sentence syntax
obj2.obj6
obj4.obj2
obj2.[[8]]
obj2=>obj1
obj1=>obj5
```

bvv2311@mail.ru

It is not possible to express truth using combinatory of the set {y, x} with two elements.
It becomes possible only when considering the set {y, f, x}, i.e., f: x → y.

Within this approach, essentially there is no difference between symbolic and neural network-based methods since graphs are represented in tensors



The form of writing the sequence (input, output) in cybernetics or, as in reinforcement learning,(state, action (input)) is typical and common

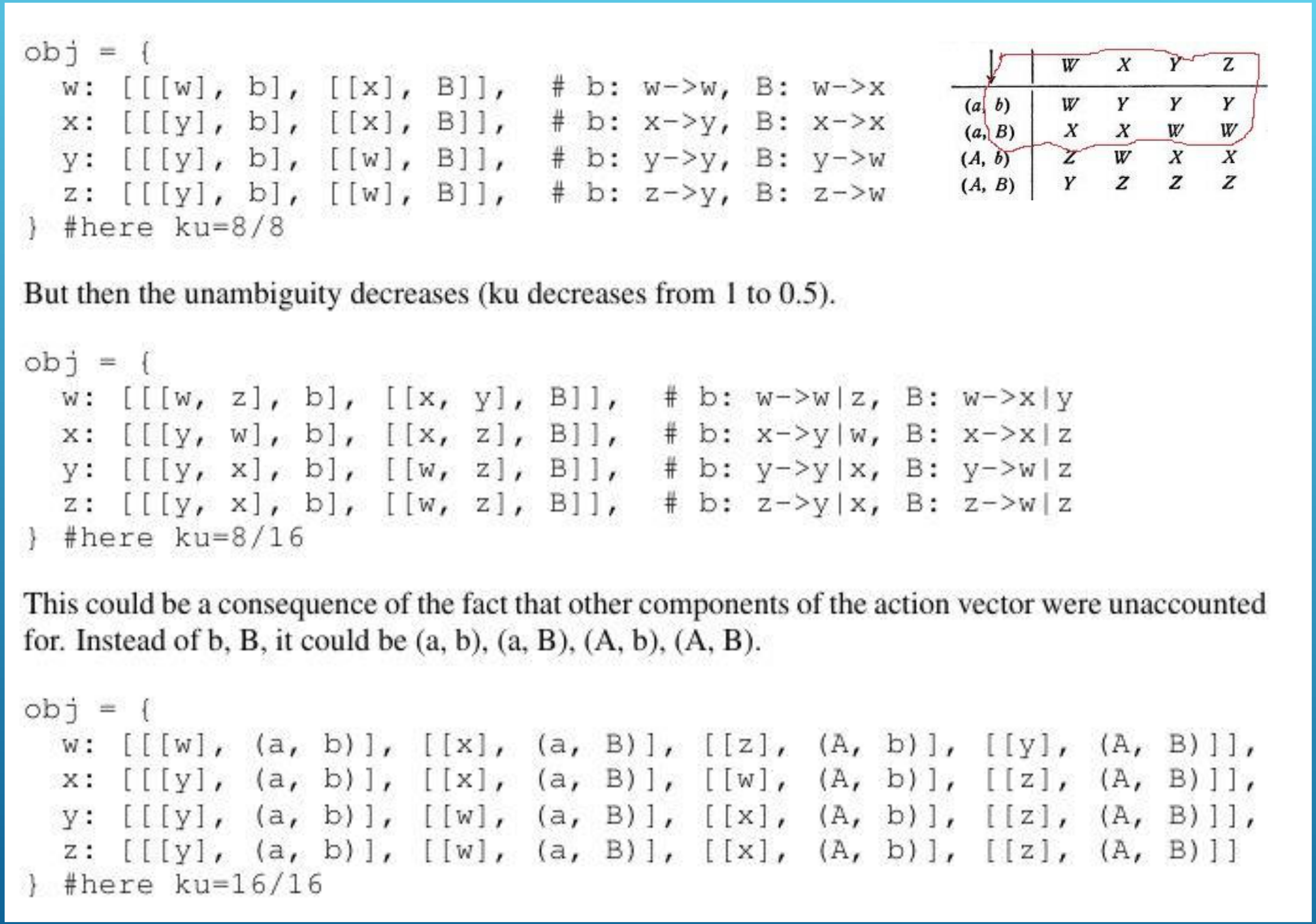
```
 [(a,g), (a,j), (a,f), (a,f), (a,f), (b,f), (b,h), (b,h), (a,h), (a,j), (b,f), (a,h), (b,j), (b,f), (a,h), (b,j), (a,f)]
```

Such a sequence can represent both a psychiatric illness (with questions a, b and answers f, g, h, j), and a description of the electrical chains Ashby (1959).

Such a protocol can be represented by a table or graph: Graph display format:

```
                      state: [[[nextState, ..], action], ..]

obj1 = {# such that its transitions:
 f: [[[f], a], [[h], b]], # a: f->f, b: f->h
 g: [[[j], a]],           # a: g->j
 h: [[[j], a], [[h], b]], # a: h->j, b: h->h
 j: [[[f], a], [[f], b]]  # a: j->f, b: j->f
} #ku=7/7
```



```
obj = {
  w: [[[w], b], [[x], B]],   # b: w->w, B: w->x
  x: [[[y], b], [[x], B]],   # b: x->y, B: x->x
  y: [[[y], b], [[w], B]],   # b: y->y, B: y->w
  z: [[[y], b], [[w], B]],   # b: z->y, B: z->w
} #here ku=8/8
```

↓	*W*	*X*	*Y*	*Z*
(*a*, *b*)	*W*	*Y*	*Y*	*Y*
(*a*, *B*)	*X*	*X*	*W*	*W*
(*A*, *b*)	*Z*	*W*	*X*	*X*
(*A*, *B*)	*Y*	*Z*	*Z*	*Z*

But then the unambiguity decreases (ku decreases from 1 to 0.5).

```
obj = {
  w: [[[w, z], b], [[x, y], B]],   # b: w->w|z, B: w->x|y
  x: [[[y, w], b], [[x, z], B]],   # b: x->y|w, B: x->x|z
  y: [[[y, x], b], [[w, z], B]],   # b: y->y|x, B: y->w|z
  z: [[[y, x], b], [[w, z], B]],   # b: z->y|x, B: z->w|z
} #here ku=8/16
```

This could be a consequence of the fact that other components of the action vector were unaccounted for. Instead of b, B, it could be (a, b), (a, B), (A, b), (A, B).

```
obj = {
  w: [[[w], (a, b)], [[x], (a, B)], [[z], (A, b)], [[y], (A, B)]],
  x: [[[y], (a, b)], [[x], (a, B)], [[w], (A, b)], [[z], (A, B)]],
  y: [[[y], (a, b)], [[w], (a, B)], [[x], (A, b)], [[z], (A, B)]],
  z: [[[y], (a, b)], [[w], (a, B)], [[x], (A, b)], [[z], (A, B)]]
} #here ku=16/16
```

The increase in ambiguity serves as a signal for redefinition.

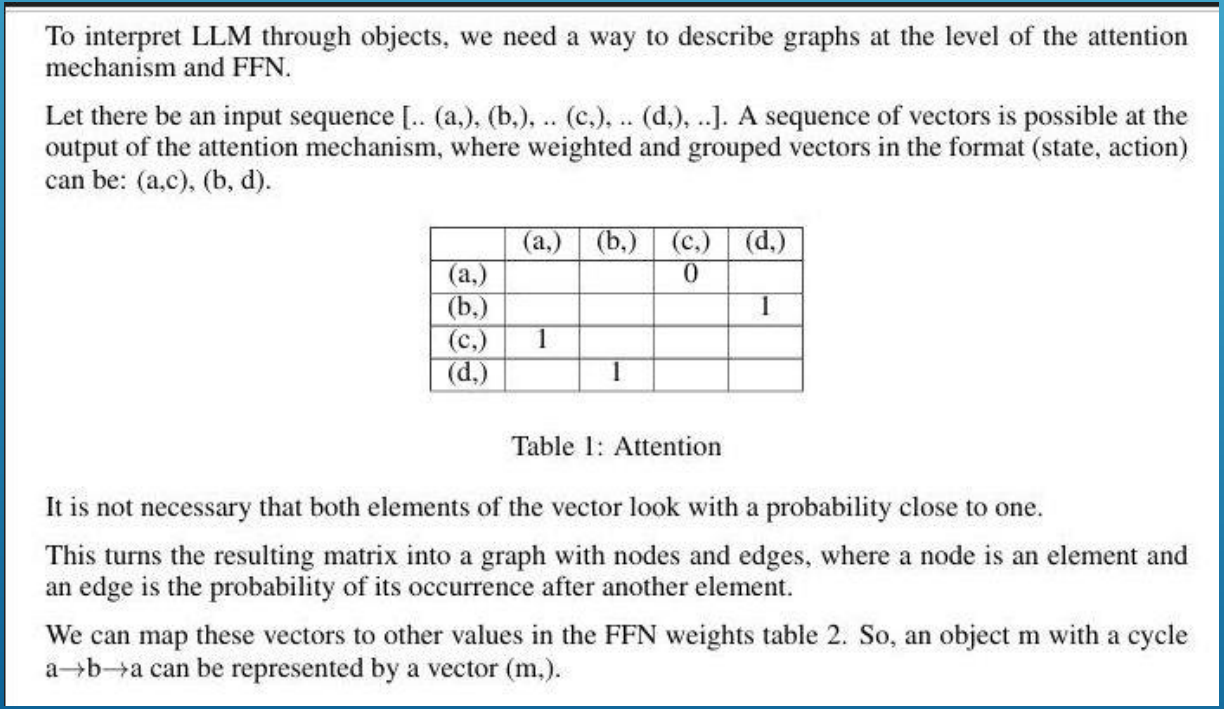
The assessment is carried out through the unambiguity coefficient ku,

which is defined as a measure of predictability (the ratio of unambiguous transitions <f,x> to all transitions <y,f,x> in a directed graph).

In transformers, the implicit teacher (due to their architecture) is entropy minimization.

After training, directed graphs with low-entropy transitions are formed.

As a multidimensional vector X traverses such a graph, it predicts the next element in the sequence



To interpret LLM through objects, we need a way to describe graphs at the level of the attention mechanism and FFN.

Let there be an input sequence [.. (a,), (b,), .. (c,), .. (d,), ..]. A sequence of vectors is possible at the output of the attention mechanism, where weighted and grouped vectors in the format (state, action) can be: (a,c), (b, d).

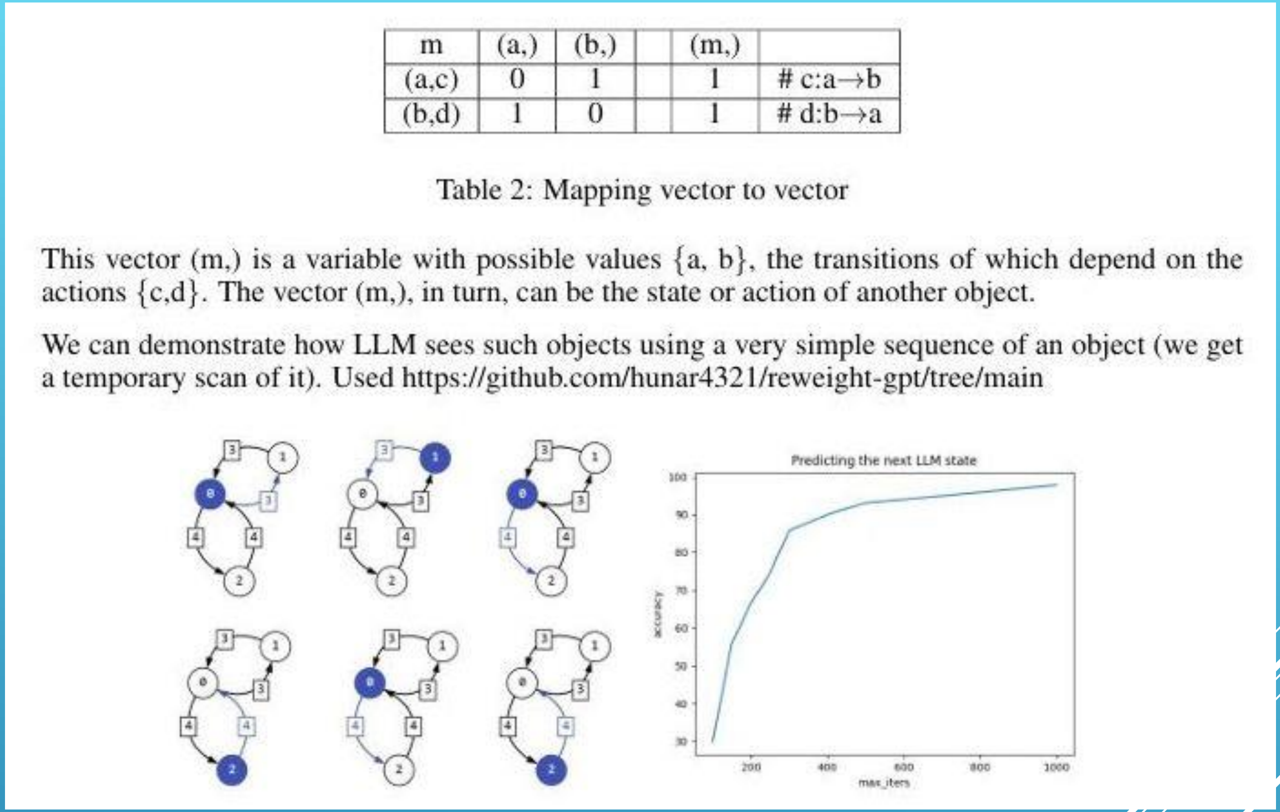
	(a,)	(b,)	(c,)	(d,)
(a,)			0	
(b,)				1
(c,)	1			
(d,)		1		

Table 1: Attention

It is not necessary that both elements of the vector look with a probability close to one.

This turns the resulting matrix into a graph with nodes and edges, where a node is an element and an edge is the probability of its occurrence after another element.

We can map these vectors to other values in the FFN weights table 2. So, an object m with a cycle a→b→a can be represented by a vector (m,).

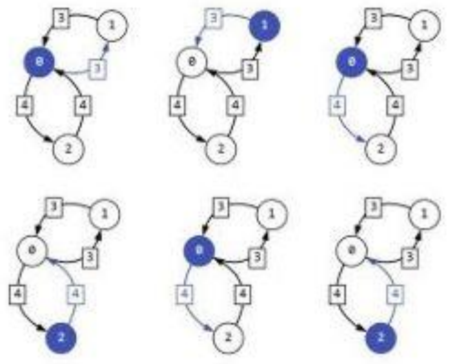


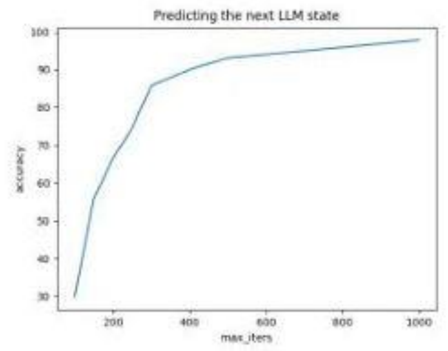
m	(a,)	(b,)		(m,)	
(a,c)	0	1		1	# c:a→b
(b,d)	1	0		1	# d:b→a

Table 2: Mapping vector to vector

This vector (m,) is a variable with possible values {a, b}, the transitions of which depend on the actions {c,d}. The vector (m,), in turn, can be the state or action of another object.

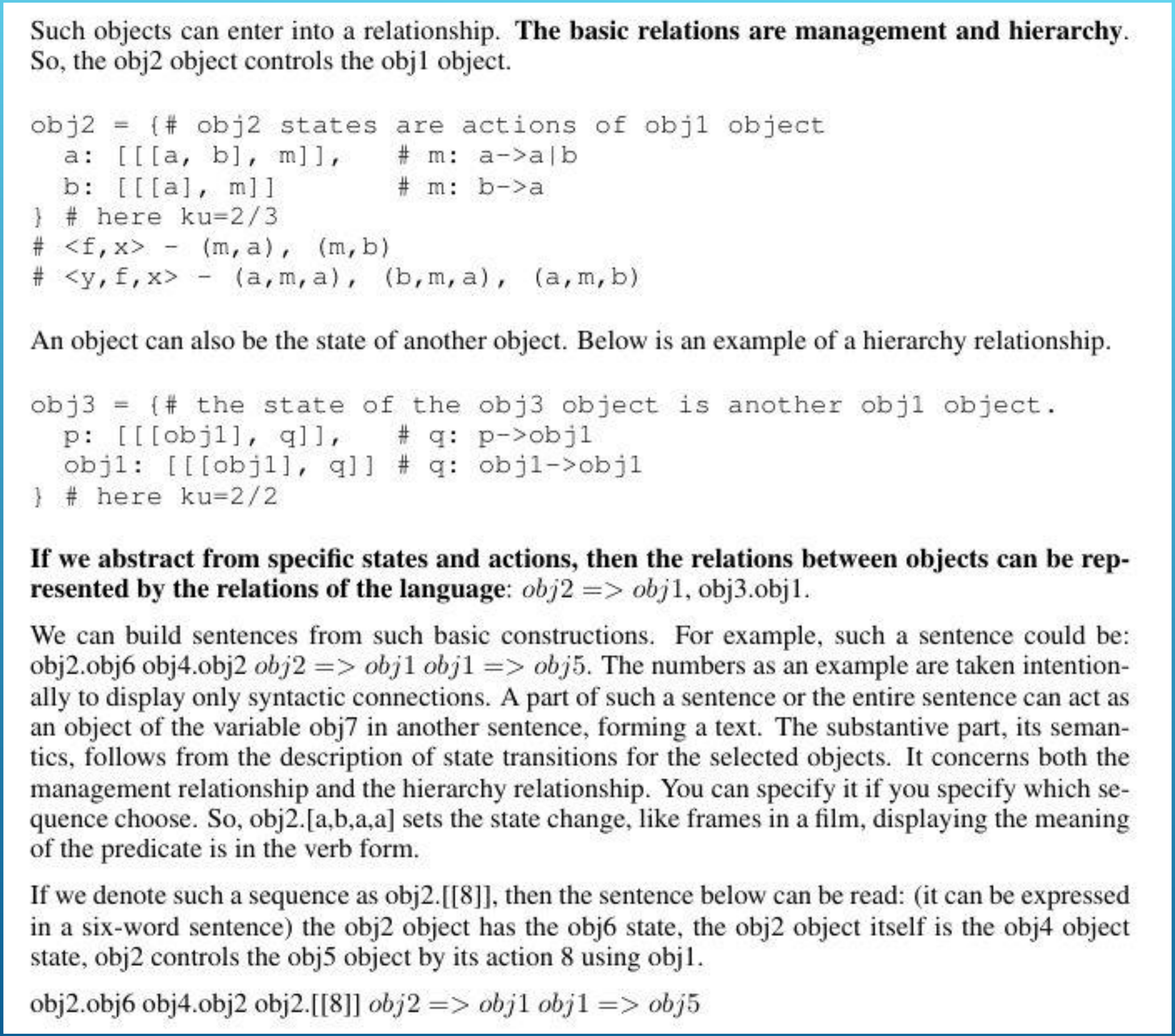
We can demonstrate how LLM sees such objects using a very simple sequence of an object (we get a temporary scan of it). Used https://github.com/hunar4321/reweight-gpt/tree/main





If the sequence of actions is excluded from consideration or incorrectly defined, then accuracy drops from 100% to 70%. This is quite understandable: after state 1, state 0 always follows, after state 2, state 0 also always follows, but after state 0, the probability that either state 1 or state 2 will follow is 50%.
seq =0, _, 1, _, 0, _, 2, _, 0, _, 2, _, 0, _, 1, _, 0, _, 2, _, 0, _, 1, ...

If the value of n_embd is reduced from 3 to 2, accuracy decreases from 100% to 10%. This drop is caused by the fact that the channel capacity (from layer to layer in the transformer) cannot provide the necessary diversity for 3 elements to transmit the function f: x->y



Such objects can enter into a relationship. **The basic relations are management and hierarchy.** So, the obj2 object controls the obj1 object.

```
obj2 = {# obj2 states are actions of obj1 object
  a: [[[a, b], m]],    # m: a->a|b
  b: [[[a], m]]        # m: b->a
} # here ku=2/3
# <f,x> - (m,a), (m,b)
# <y,f,x> - (a,m,a), (b,m,a), (a,m,b)
```

An object can also be the state of another object. Below is an example of a hierarchy relationship.

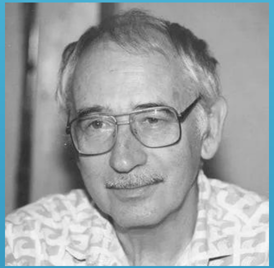
```
obj3 = {# the state of the obj3 object is another obj1 object.
  p: [[[obj1], q]],    # q: p->obj1
  obj1: [[[obj1], q]] # q: obj1->obj1
} # here ku=2/2
```

If we abstract from specific states and actions, then the relations between objects can be represented by the relations of the language: $obj2 => obj1$, obj3.obj1.

We can build sentences from such basic constructions. For example, such a sentence could be: obj2.obj6 obj4.obj2 $obj2 => obj1$ $obj1 => obj5$. The numbers as an example are taken intentionally to display only syntactic connections. A part of such a sentence or the entire sentence can act as an object of the variable obj7 in another sentence, forming a text. The substantive part, its semantics, follows from the description of state transitions for the selected objects. It concerns both the management relationship and the hierarchy relationship. You can specify it if you specify which sequence choose. So, obj2.[a,b,a,a] sets the state change, like frames in a film, displaying the meaning of the predicate is in the verb form.

If we denote such a sequence as obj2.[[8]], then the sentence below can be read: (it can be expressed in a six-word sentence) the obj2 object has the obj6 state, the obj2 object itself is the obj4 object state, obj2 controls the obj5 object by its action 8 using obj1.

obj2.obj6 obj4.obj2 obj2.[[8]] $obj2 => obj1$ $obj1 => obj5$



Sirius International Mathematics Center, Sirius
March 24-28, 2025